МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

НГТУ НЭТИ | Факультет прикладной математики и информатики

Кафедра прикладной математики

Практическое задание № 1

по дисциплине «Уравнения математической физики»

РЕШЕНИЕ ЭЛЛИПТИЧЕСКИХ КРАЕВЫХ ЗАДАЧ

МЕТОДОМ КОНЕЧНЫХ РАЗНОСТЕЙ

Бригада 1	ИСАКИН ДАНИИЛ
Группа ПМ-13	ВОСТРЕЦОВА ЕКАТЕРИНА
Вариант 6	

Преподаватели	ЗАДОРОЖНЫЙ АЛЕКСАНДР ГЕННАДЬЕВИЧ
	ЛЕОНОВИЧ ДАРЬЯНА

Новосибирск, 2024

1. Цель работы

Разработать программу решения эллиптической краевой задачи методом конечных разностей. Протестировать программу и численно оценить порядок аппроксимации.

2. Задание

Уравнение: $-\operatorname{div}(\lambda \operatorname{grad} u) + \gamma u = f$ для функции $u = u(x, y)$, краевые условия: $u|_{s_1} = u_g$, $\lambda \frac{\partial u}{\partial n}\Big|_{s_2} = \theta$, Область Ω имеет L-образную форму

3. Анализ

$$-\lambda\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + \gamma u = f$$

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

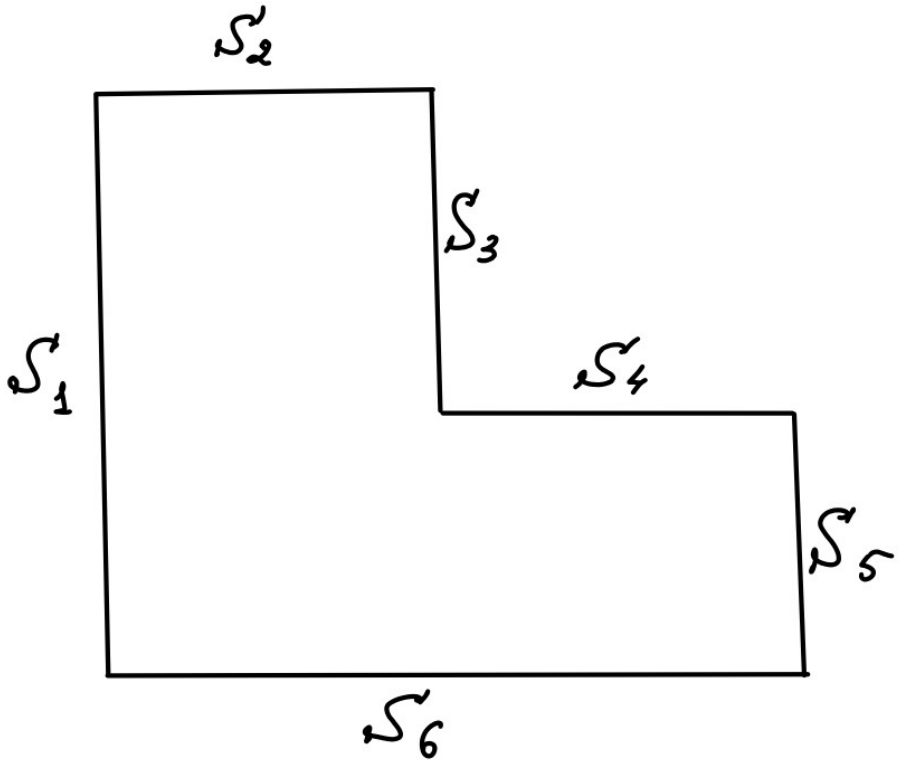
$$-\lambda \Delta u + \gamma u = f$$

Дискретный аналог оператора Лапласа на пятиточечном шаблоне:

$$\Delta_h u_{i,j} = \frac{2u_{i-1,j}}{h^x_{i-1}\left(h^x_i + h^x_{i-1}\right)} + \frac{2u_{i,j-1}}{h^y_{j-1}\left(h^y_j + h^y_{j-1}\right)} + \frac{2u_{i+1,j}}{h^x_i\left(h^x_i + h^x_{i-1}\right)} + \frac{2u_{i,j+1}}{h^y_j\left(h^y_j + h^x_{j-1}\right)} - \left(\frac{2}{h^x_{i-1}h^x_i} + \frac{2}{h^x_{j-1}h^y_j}\right)u_{i,j}$$

Построим сетку, узлы нумеруем снизу-вверх и слева направо. Для решения поставленной задачи из линий сетки получаем координаты узлов. Матрица формируется одним проходом по всем узлам, для регулярных узлов заполняется согласно пятиточечному шаблону, для прочих – в соответствии с краевыми условиями. Матрица хранится как пятидиагональная

Форма области:



4. Исследования

Тест №1 (Полином первой степени)

u(x,y) = x+y
λ=1, γ=1
f(x,y) = x+y
Краевые условия 1-ого рода на всех сторонах
Содержание файла CalcArea.txt (Формат описан ниже)
Абсолютна погрешность: $\|\widetilde{U} - U(h)\|$=2.66454e-15

```
3 3
0 0 5 0 10 0
0 5 5 5 10 5
0 10 5 10 10 10
2
1 1 2 1 3
1 2 3 1 2
6
1 1 1 1 1 3
2 1 1 2 3 3
3 1 2 2 2 3
4 1 2 3 2 2
5 1 3 3 1 2
6 1 1 3 1 1
2 1 2 1
2 1 2 1
```

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
1	2.500	0.000	2.500000e+00	2.500000e+00	0.000000e+00
2	5.000	0.000	5.000000e+00	5.000000e+00	0.000000e+00
3	7.500	0.000	7.500000e+00	7.500000e+00	0.000000e+00
4	10.000	0.000	1.000000e+01	1.000000e+01	0.000000e+00
5	0.000	2.500	2.500000e+00	2.500000e+00	0.000000e+00
* 6	2.500	2.500	5.000000e+00	5.000000e+00	2.575717e-14
* 7	5.000	2.500	7.500000e+00	7.500000e+00	4.440892e-15
* 8	7.500	2.500	1.000000e+01	1.000000e+01	1.776357e-15
9	10.000	2.500	1.250000e+01	1.250000e+01	0.000000e+00
10	0.000	5.000	5.000000e+00	5.000000e+00	0.000000e+00
* 11	2.500	5.000	7.500000e+00	7.500000e+00	4.440892e-15
12	5.000	5.000	1.000000e+01	1.000000e+01	0.000000e+00
13	7.500	5.000	1.250000e+01	1.250000e+01	0.000000e+00
14	10.000	5.000	1.500000e+01	1.500000e+01	0.000000e+00
15	0.000	7.500	7.500000e+00	7.500000e+00	0.000000e+00
* 16	2.500	7.500	1.000000e+01	1.000000e+01	1.776357e-15
17	5.000	7.500	1.250000e+01	1.250000e+01	0.000000e+00
20	0.000	10.000	1.000000e+01	1.000000e+01	0.000000e+00
21	2.500	10.000	1.250000e+01	1.250000e+01	0.000000e+00
22	5.000	10.000	1.500000e+01	1.500000e+01	0.000000e+00

Тест №2 (Полином второй степени)

$u(x,y) = x^2+y^2$

$\lambda=1$, $\gamma=1$

$f(x,y) = -4+x^2+y^2$

Краевые условия 1-ого рода на всех границах

Содержание файла CalcArea.txt такое же как в Тесте №1

Абсолютна погрешность: $\|\widetilde{U} - U(h)\|$=1.24345e-14

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
1	2.500	0.000	6.250000e+00	6.250000e+00	0.000000e+00
2	5.000	0.000	2.500000e+01	2.500000e+01	0.000000e+00
3	7.500	0.000	5.625000e+01	5.625000e+01	0.000000e+00
4	10.000	0.000	1.000000e+02	1.000000e+02	0.000000e+00
5	0.000	2.500	6.250000e+00	6.250000e+00	0.000000e+00
* 6	2.500	2.500	1.250000e+01	1.250000e+01	1.847411e-13
* 7	5.000	2.500	3.125000e+01	3.125000e+01	2.842171e-14
* 8	7.500	2.500	6.250000e+01	6.250000e+01	7.105427e-15
9	10.000	2.500	1.062500e+02	1.062500e+02	0.000000e+00
10	0.000	5.000	2.500000e+01	2.500000e+01	0.000000e+00
* 11	2.500	5.000	3.125000e+01	3.125000e+01	2.842171e-14
12	5.000	5.000	5.000000e+01	5.000000e+01	0.000000e+00
13	7.500	5.000	8.125000e+01	8.125000e+01	0.000000e+00
14	10.000	5.000	1.250000e+02	1.250000e+02	0.000000e+00
15	0.000	7.500	5.625000e+01	5.625000e+01	0.000000e+00
* 16	2.500	7.500	6.250000e+01	6.250000e+01	7.105427e-15
17	5.000	7.500	8.125000e+01	8.125000e+01	0.000000e+00
20	0.000	10.000	1.000000e+02	1.000000e+02	0.000000e+00
21	2.500	10.000	1.062500e+02	1.062500e+02	0.000000e+00
22	5.000	10.000	1.250000e+02	1.250000e+02	0.000000e+00

Тест №3 (Полином третьей степени)

$u(x,y) = x^3+y^3$

$\lambda=1$, $\gamma=1$

$f(x,y) = -6x-6y+x^3+y^3$

Краевые условия 1-ого рода на всех границах

Содержание файла CalcArea.txt такое же как в Тесте №1

Абсолютна погрешность: $\|\widetilde{U}-U(h)\|$=9.69702e-14

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
1	2.500	0.000	1.562500e+01	1.562500e+01	0.000000e+00
2	5.000	0.000	1.250000e+02	1.250000e+02	0.000000e+00
3	7.500	0.000	4.218750e+02	4.218750e+02	0.000000e+00
4	10.000	0.000	1.000000e+03	1.000000e+03	0.000000e+00
5	0.000	2.500	1.562500e+01	1.562500e+01	0.000000e+00
* 6	2.500	2.500	3.125000e+01	3.125000e+01	1.463718e-12
* 7	5.000	2.500	1.406250e+02	1.406250e+02	2.273737e-13
* 8	7.500	2.500	4.375000e+02	4.375000e+02	5.684342e-14
9	10.000	2.500	1.015625e+03	1.015625e+03	0.000000e+00
10	0.000	5.000	1.250000e+02	1.250000e+02	0.000000e+00
* 11	2.500	5.000	1.406250e+02	1.406250e+02	2.273737e-13
12	5.000	5.000	2.500000e+02	2.500000e+02	0.000000e+00
13	7.500	5.000	5.468750e+02	5.468750e+02	0.000000e+00
14	10.000	5.000	1.125000e+03	1.125000e+03	0.000000e+00
15	0.000	7.500	4.218750e+02	4.218750e+02	0.000000e+00
* 16	2.500	7.500	4.375000e+02	4.375000e+02	5.684342e-14
17	5.000	7.500	5.468750e+02	5.468750e+02	0.000000e+00
20	0.000	10.000	1.000000e+03	1.000000e+03	0.000000e+00
21	2.500	10.000	1.015625e+03	1.015625e+03	0.000000e+00
22	5.000	10.000	1.125000e+03	1.125000e+03	0.000000e+00

Тест №4 (Полином четвертой степени)

$u(x,y) = x^4+y^4$

$\lambda=1$, $\gamma=1$

$f(x,y) = -12x^2-12y^2+x^4+y^4$

Краевые условия 1-ого рода на всех границах

Содержание файла CalcArea.txt такое же как в Тесте №1

Абсолютна погрешность: $\|\widetilde{U}-U(h)\|=40.5432$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
1	2.500	0.000	3.906250e+01	3.906250e+01	0.000000e+00
2	5.000	0.000	6.250000e+02	6.250000e+02	0.000000e+00
3	7.500	0.000	3.164062e+03	3.164062e+03	0.000000e+00
4	10.000	0.000	1.000000e+04	1.000000e+04	0.000000e+00
5	0.000	2.500	3.906250e+01	3.906250e+01	0.000000e+00
* 6	2.500	2.500	9.702819e+01	7.812500e+01	1.890319e+01
* 7	5.000	2.500	6.828163e+02	6.640625e+02	1.875383e+01
* 8	7.500	2.500	3.220199e+03	3.203125e+03	1.707354e+01
9	10.000	2.500	1.003906e+04	1.003906e+04	0.000000e+00
10	0.000	5.000	6.250000e+02	6.250000e+02	0.000000e+00
* 11	2.500	5.000	6.828163e+02	6.640625e+02	1.875383e+01
12	5.000	5.000	1.250000e+03	1.250000e+03	0.000000e+00
13	7.500	5.000	3.789062e+03	3.789062e+03	0.000000e+00
14	10.000	5.000	1.062500e+04	1.062500e+04	0.000000e+00
15	0.000	7.500	3.164062e+03	3.164062e+03	0.000000e+00
* 16	2.500	7.500	3.220199e+03	3.203125e+03	1.707354e+01
17	5.000	7.500	3.789062e+03	3.789062e+03	0.000000e+00
20	0.000	10.000	1.000000e+04	1.000000e+04	0.000000e+00
21	2.500	10.000	1.003906e+04	1.003906e+04	0.000000e+00
22	5.000	10.000	1.062500e+04	1.062500e+04	0.000000e+00

Произведем дробление сетки

Дробление в 2 раза

Параметры дробления

OX: n = 4 q = 1 n = 4 q = 1

OY: n = 4 q = 1 n = 4 q = 1

Абсолютна погрешность: $\|\widetilde{U}-U(\frac{h}{2})\|=11.119$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
2	2.500	0.000	3.906250e+01	3.906250e+01	0.000000e+00
4	5.000	0.000	6.250000e+02	6.250000e+02	0.000000e+00
6	7.500	0.000	3.164062e+03	3.164062e+03	0.000000e+00
8	10.000	0.000	1.000000e+04	1.000000e+04	0.000000e+00
18	0.000	2.500	3.906250e+01	3.906250e+01	0.000000e+00
* 20	2.500	2.500	8.321735e+01	7.812500e+01	5.092346e+00
* 22	5.000	2.500	6.692039e+02	6.640625e+02	5.141438e+00
* 24	7.500	2.500	3.207859e+03	3.203125e+03	4.734493e+00
26	10.000	2.500	1.003906e+04	1.003906e+04	0.000000e+00
36	0.000	5.000	6.250000e+02	6.250000e+02	0.000000e+00
* 38	2.500	5.000	6.692039e+02	6.640625e+02	5.141438e+00
40	5.000	5.000	1.250000e+03	1.250000e+03	0.000000e+00
42	7.500	5.000	3.789062e+03	3.789062e+03	0.000000e+00
44	10.000	5.000	1.062500e+04	1.062500e+04	0.000000e+00
54	0.000	7.500	3.164062e+03	3.164062e+03	0.000000e+00
* 56	2.500	7.500	3.207859e+03	3.203125e+03	4.734493e+00
58	5.000	7.500	3.789062e+03	3.789062e+03	0.000000e+00
72	0.000	10.000	1.000000e+04	1.000000e+04	0.000000e+00
74	2.500	10.000	1.003906e+04	1.003906e+04	0.000000e+00
76	5.000	10.000	1.062500e+04	1.062500e+04	0.000000e+00

Деление в 4 раза

Параметры дробления
OX: n = 8 q = 1 n = 8 q = 1
OY: n = 8 q = 1 n = 8 q = 1

Абсолютна погрешность: $\|\widetilde{U} - U(\frac{h}{4})\| = 2.86223$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
4	2.500	0.000	3.906250e+01	3.906250e+01	0.000000e+00
8	5.000	0.000	6.250000e+02	6.250000e+02	0.000000e+00
12	7.500	0.000	3.164062e+03	3.164062e+03	0.000000e+00
16	10.000	0.000	1.000000e+04	1.000000e+04	0.000000e+00
68	0.000	2.500	3.906250e+01	3.906250e+01	0.000000e+00
* 72	2.500	2.500	7.942768e+01	7.812500e+01	1.302678e+00
* 76	5.000	2.500	6.653865e+02	6.640625e+02	1.324023e+00
* 80	7.500	2.500	3.204348e+03	3.203125e+03	1.222559e+00
84	10.000	2.500	1.003906e+04	1.003906e+04	0.000000e+00
136	0.000	5.000	6.250000e+02	6.250000e+02	0.000000e+00
* 140	2.500	5.000	6.653865e+02	6.640625e+02	1.324023e+00
144	5.000	5.000	1.250000e+03	1.250000e+03	0.000000e+00
148	7.500	5.000	3.789062e+03	3.789062e+03	0.000000e+00
152	10.000	5.000	1.062500e+04	1.062500e+04	0.000000e+00
204	0.000	7.500	3.164062e+03	3.164062e+03	0.000000e+00
* 208	2.500	7.500	3.204348e+03	3.203125e+03	1.222559e+00
212	5.000	7.500	3.789062e+03	3.789062e+03	0.000000e+00
272	0.000	10.000	1.000000e+04	1.000000e+04	0.000000e+00
276	2.500	10.000	1.003906e+04	1.003906e+04	0.000000e+00
280	5.000	10.000	1.062500e+04	1.062500e+04	0.000000e+00

Оценим порядок сходимости:

$$\log_2\left(\frac{\|\widetilde{U}-U(h)\|}{\|\widetilde{U}-U(\frac{h}{2})\|}\right)=\frac{40.5432}{11.119}=1.86643$$

$$\log_2\left(\frac{\|\widetilde{U}-U(\frac{h}{2})\|}{\|\widetilde{U}-U(\frac{h}{4})\|}\right)=\frac{11.119}{2.86223}=1.95782$$

Тест №5 (На 2 КУ, полином первой степени)

u(x,y) = x+y

λ=1, γ=1

f(x,y) = x+y

Краевые условия 2-го рода на нижней границе S6 и S3

Краевые условия 1-ого рода на всех остальных сторонах

Содержание файла CalcArea.txt (Формат описан ниже)

Абсолютна погрешность: $\|\widetilde{U} - U(h)\| = 7.33756e\text{-}15$

```
3 3
0 0 5 0 10 0
0 5 5 5 10 5
0 10 5 10 10 10
2
1 1 2 1 3
1 2 3 1 2
6
1 1 1 1 1 3
2 1 1 2 3 3
3 2 2 2 2 3
4 1 2 3 2 2
5 1 3 3 1 2
6 2 1 3 1 1
2 1 2 1
2 1 2 1
```

Расчетная таблица. Символом * отмечены внутренние узлы сетки

	N	X	Y	U	U*	|U* - U|
	0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
	1	2.500	0.000	2.500000e+00	2.500000e+00	4.884981e-15
	2	5.000	0.000	5.000000e+00	5.000000e+00	3.552714e-15
	3	7.500	0.000	7.500000e+00	7.500000e+00	1.776357e-15
	4	10.000	0.000	1.000000e+01	1.000000e+01	0.000000e+00
	5	0.000	2.500	2.500000e+00	2.500000e+00	0.000000e+00
*	6	2.500	2.500	5.000000e+00	5.000000e+00	8.881784e-16
*	7	5.000	2.500	7.500000e+00	7.500000e+00	1.776357e-15
*	8	7.500	2.500	1.000000e+01	1.000000e+01	1.776357e-15
	9	10.000	2.500	1.250000e+01	1.250000e+01	0.000000e+00
	10	0.000	5.000	5.000000e+00	5.000000e+00	0.000000e+00
*	11	2.500	5.000	7.500000e+00	7.500000e+00	8.881784e-16
	12	5.000	5.000	1.000000e+01	1.000000e+01	0.000000e+00
	13	7.500	5.000	1.250000e+01	1.250000e+01	0.000000e+00
	14	10.000	5.000	1.500000e+01	1.500000e+01	0.000000e+00
	15	0.000	7.500	7.500000e+00	7.500000e+00	0.000000e+00
*	16	2.500	7.500	1.000000e+01	1.000000e+01	1.776357e-15
	17	5.000	7.500	1.250000e+01	1.250000e+01	1.776357e-15
	20	0.000	10.000	1.000000e+01	1.000000e+01	0.000000e+00
	21	2.500	10.000	1.250000e+01	1.250000e+01	0.000000e+00
	22	5.000	10.000	1.500000e+01	1.500000e+01	0.000000e+00

Тест №6 (2КУ, Полином второго порядка)

$u(x,y) = x^2+y^2$

$\lambda=1$, $\gamma=1$

$f(x,y) = -4+x^2+y^2$

Краевые условия 2-го рода на нижней границе S6 и S3

Краевые условия 1-ого рода на всех остальных сторонах

Содержание файла CalcArea.txt аналогично Тесту №5

Абсолютна погрешность: $\|\widetilde{U} - U(h)\| = 14.1277$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
1	2.500	0.000	1.328237e+01	6.250000e+00	7.032370e+00
2	5.000	0.000	3.209316e+01	2.500000e+01	7.093156e+00
3	7.500	0.000	6.326683e+01	5.625000e+01	7.016828e+00
4	10.000	0.000	1.000000e+02	1.000000e+02	0.000000e+00
5	0.000	2.500	6.250000e+00	6.250000e+00	0.000000e+00
* 6	2.500	2.500	1.328237e+01	1.250000e+01	7.823699e-01
* 7	5.000	2.500	3.209316e+01	3.125000e+01	8.431565e-01
* 8	7.500	2.500	6.326683e+01	6.250000e+01	7.668277e-01
9	10.000	2.500	1.062500e+02	1.062500e+02	0.000000e+00
10	0.000	5.000	2.500000e+01	2.500000e+01	0.000000e+00
* 11	2.500	5.000	3.139376e+01	3.125000e+01	1.437646e-01
12	5.000	5.000	5.000000e+01	5.000000e+01	0.000000e+00
13	7.500	5.000	8.125000e+01	8.125000e+01	0.000000e+00
14	10.000	5.000	1.250000e+02	1.250000e+02	0.000000e+00
15	0.000	7.500	5.625000e+01	5.625000e+01	0.000000e+00
* 16	2.500	7.500	6.319122e+01	6.250000e+01	6.912178e-01
17	5.000	7.500	8.819122e+01	8.125000e+01	6.941218e+00
20	0.000	10.000	1.000000e+02	1.000000e+02	0.000000e+00
21	2.500	10.000	1.062500e+02	1.062500e+02	0.000000e+00
22	5.000	10.000	1.250000e+02	1.250000e+02	0.000000e+00

Произведем дробление

Дробление в 2 раза

Параметры дробления

OX: n = 4 q = 1 n = 4 q = 1

OY: n = 4 q = 1 n = 4 q = 1

Абсолютна погрешность: $\|\widetilde{U} - U(\frac{h}{2})\| = 4.43239$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
2	2.500	0.000	8.458154e+00	6.250000e+00	2.208154e+00
4	5.000	0.000	2.724843e+01	2.500000e+01	2.248429e+00
6	7.500	0.000	5.845642e+01	5.625000e+01	2.206416e+00
8	10.000	0.000	1.000000e+02	1.000000e+02	0.000000e+00
18	0.000	2.500	6.250000e+00	6.250000e+00	0.000000e+00
* 20	2.500	2.500	1.268991e+01	1.250000e+01	1.899059e-01
* 22	5.000	2.500	3.145857e+01	3.125000e+01	2.085704e-01
* 24	7.500	2.500	6.268464e+01	6.250000e+01	1.846374e-01
26	10.000	2.500	1.062500e+02	1.062500e+02	0.000000e+00
36	0.000	5.000	2.500000e+01	2.500000e+01	0.000000e+00
* 38	2.500	5.000	3.131482e+01	3.125000e+01	6.481844e-02
40	5.000	5.000	5.000000e+01	5.000000e+01	0.000000e+00
42	7.500	5.000	8.125000e+01	8.125000e+01	0.000000e+00
44	10.000	5.000	1.250000e+02	1.250000e+02	0.000000e+00
54	0.000	7.500	5.625000e+01	5.625000e+01	0.000000e+00
* 56	2.500	7.500	6.266488e+01	6.250000e+01	1.648836e-01
58	5.000	7.500	8.341826e+01	8.125000e+01	2.168260e+00
72	0.000	10.000	1.000000e+02	1.000000e+02	0.000000e+00
74	2.500	10.000	1.062500e+02	1.062500e+02	0.000000e+00
76	5.000	10.000	1.250000e+02	1.250000e+02	0.000000e+00

Деление в 4 раза

Параметры дробления
OX: n = 8 q = 1 n = 8 q = 1
OY: n = 8 q = 1 n = 8 q = 1

Абсолютна погрешность: $\|\widetilde{U} - U(\frac{h}{4})\| = 1.66213$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
4	2.500	0.000	7.077861e+00	6.250000e+00	8.278611e-01
8	5.000	0.000	2.584757e+01	2.500000e+01	8.475672e-01
12	7.500	0.000	5.707742e+01	5.625000e+01	8.274179e-01
16	10.000	0.000	1.000000e+02	1.000000e+02	0.000000e+00
68	0.000	2.500	6.250000e+00	6.250000e+00	0.000000e+00
* 72	2.500	2.500	1.256439e+01	1.250000e+01	6.439474e-02
* 76	5.000	2.500	3.132261e+01	3.125000e+01	7.260792e-02
* 80	7.500	2.500	6.256193e+01	6.250000e+01	6.192974e-02
84	10.000	2.500	1.062500e+02	1.062500e+02	0.000000e+00
136	0.000	5.000	2.500000e+01	2.500000e+01	0.000000e+00
* 140	2.500	5.000	3.127942e+01	3.125000e+01	2.942237e-02
144	5.000	5.000	5.000000e+01	5.000000e+01	0.000000e+00
148	7.500	5.000	8.125000e+01	8.125000e+01	0.000000e+00
152	10.000	5.000	1.250000e+02	1.250000e+02	0.000000e+00
204	0.000	7.500	5.625000e+01	5.625000e+01	0.000000e+00
* 208	2.500	7.500	6.255475e+01	6.250000e+01	5.474650e-02
212	5.000	7.500	8.206070e+01	8.125000e+01	8.106975e-01
272	0.000	10.000	1.000000e+02	1.000000e+02	0.000000e+00
276	2.500	10.000	1.062500e+02	1.062500e+02	0.000000e+00
280	5.000	10.000	1.250000e+02	1.250000e+02	0.000000e+00

Деление в 8 раз

Параметры дробления
OX: n = 16 q = 1 n = 16 q = 1
OY: n = 16 q = 1 n = 16 q = 1
Абсолютна погрешность: $\|\widetilde{U} - U(\frac{h}{8})\| = 0.711643$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
8	2.500	0.000	6.604314e+00	6.250000e+00	3.543139e-01
16	5.000	0.000	2.536394e+01	2.500000e+01	3.639404e-01
24	7.500	0.000	5.660415e+01	5.625000e+01	3.541521e-01
32	10.000	0.000	1.000000e+02	1.000000e+02	0.000000e+00
264	0.000	2.500	6.250000e+00	6.250000e+00	0.000000e+00
* 272	2.500	2.500	1.252678e+01	1.250000e+01	2.677874e-02
* 280	5.000	2.500	3.128074e+01	3.125000e+01	3.073986e-02
* 288	7.500	2.500	6.252560e+01	6.250000e+01	2.560137e-02
296	10.000	2.500	1.062500e+02	1.062500e+02	0.000000e+00
528	0.000	5.000	2.500000e+01	2.500000e+01	0.000000e+00
* 536	2.500	5.000	3.126316e+01	3.125000e+01	1.316255e-02
544	5.000	5.000	5.000000e+01	5.000000e+01	0.000000e+00
552	7.500	5.000	8.125000e+01	8.125000e+01	0.000000e+00
560	10.000	5.000	1.250000e+02	1.250000e+02	0.000000e+00
792	0.000	7.500	5.625000e+01	5.625000e+01	0.000000e+00
* 800	2.500	7.500	6.252248e+01	6.250000e+01	2.248329e-02
808	5.000	7.500	8.159645e+01	8.125000e+01	3.464464e-01
1056	0.000	10.000	1.000000e+02	1.000000e+02	0.000000e+00
1064	2.500	10.000	1.062500e+02	1.062500e+02	0.000000e+00
1072	5.000	10.000	1.250000e+02	1.250000e+02	0.000000e+00

Оценим порядок сходимости:

$$\log_2\left(\frac{\|\widetilde{U} - U(h)\|}{\|\widetilde{U} - U(\frac{h}{2})\|}\right) = \frac{14.1277}{4.43239} = 1.6724$$

$$\log_2\left(\frac{\|\widetilde{U} - U(\frac{h}{2})\|}{\|\widetilde{U} - U(\frac{h}{4})\|}\right) = \frac{4.43239}{1.66213} = 1.41505$$

$$\log_2\left(\frac{\|\widetilde{U} - U(\frac{h}{4})\|}{\|\widetilde{U} - U(\frac{h}{8})\|}\right) = \frac{1.66213}{0.711643} = 1.22381$$

Тест №6 на порядок сходимости на равномерной сетке

$u=\sin\frac{\pi x}{2}+\cos\frac{\pi y}{2}, f=\left(\frac{\pi^2}{4}+1\right)*\left(\sin\frac{\pi x}{2}+\cos\frac{\pi y}{2}\right), \gamma=1, \lambda=1$

Краевые условия 1-ого типа на всех границах

Содержание файла CalcArea.txt такое же как в Тесте №1

Абсолютна погрешность: $\|\widetilde{U}-U(h)\|=2.61644$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	1.000000e+00	1.000000e+00	1.986189e-13
1	2.500	0.000	2.928932e-01	2.928932e-01	5.817569e-14
2	5.000	0.000	2.000000e+00	2.000000e+00	3.972378e-13
3	7.500	0.000	2.928932e-01	2.928932e-01	5.817569e-14
4	10.000	0.000	1.000000e+00	1.000000e+00	1.986189e-13
5	0.000	2.500	-7.071068e-01	-7.071068e-01	1.404432e-13
* 6	2.500	2.500	-3.168275e+00	-1.414214e+00	1.754061e+00
* 7	5.000	2.500	3.033424e-01	2.928932e-01	1.044913e-02
* 8	7.500	2.500	-3.069831e+00	-1.414214e+00	1.655617e+00
9	10.000	2.500	-7.071068e-01	-7.071068e-01	1.404432e-13
10	0.000	5.000	3.061617e-16	3.061617e-16	6.079159e-29
* 11	2.500	5.000	-1.716157e+00	-7.071068e-01	1.009050e+00
12	5.000	5.000	1.000000e+00	1.000000e+00	1.986189e-13
13	7.500	5.000	-7.071068e-01	-7.071068e-01	1.404432e-13
14	10.000	5.000	9.184851e-16	9.184851e-16	1.824241e-28
15	0.000	7.500	7.071068e-01	7.071068e-01	1.404432e-13
* 16	2.500	7.500	-9.844394e-02	-8.881784e-16	9.844394e-02
17	5.000	7.500	1.707107e+00	1.707107e+00	3.390621e-13
20	0.000	10.000	-1.000000e+00	-1.000000e+00	1.986189e-13
21	2.500	10.000	-1.707107e+00	-1.707107e+00	3.390621e-13
22	5.000	10.000	0.000000e+00	0.000000e+00	0.000000e+00

Дробление сетки на 2

Параметры дробления

OX: n = 4 q = 1 n = 4 q = 1

OY: n = 4 q = 1 n = 4 q = 1

Абсолютна погрешность: $\|\widetilde{U}-U(\frac{h}{2})\|=0.553959$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	1.000000e+00	1.000000e+00	0.000000e+00
2	2.500	0.000	2.928932e-01	2.928932e-01	0.000000e+00
4	5.000	0.000	2.000000e+00	2.000000e+00	0.000000e+00
6	7.500	0.000	2.928932e-01	2.928932e-01	0.000000e+00
8	10.000	0.000	1.000000e+00	1.000000e+00	0.000000e+00
18	0.000	2.500	-7.071068e-01	-7.071068e-01	0.000000e+00
* 20	2.500	2.500	-1.784102e+00	-1.414214e+00	3.698884e-01
* 22	5.000	2.500	3.273159e-01	2.928932e-01	3.442273e-02
* 24	7.500	2.500	-1.776833e+00	-1.414214e+00	3.626197e-01
26	10.000	2.500	-7.071068e-01	-7.071068e-01	0.000000e+00
36	0.000	5.000	3.061617e-16	3.061617e-16	0.000000e+00
* 38	2.500	5.000	-9.002965e-01	-7.071068e-01	1.931897e-01
40	5.000	5.000	1.000000e+00	1.000000e+00	0.000000e+00
42	7.500	5.000	-7.071068e-01	-7.071068e-01	0.000000e+00
44	10.000	5.000	9.184851e-16	9.184851e-16	0.000000e+00
54	0.000	7.500	7.071068e-01	7.071068e-01	0.000000e+00
* 56	2.500	7.500	-7.268693e-03	-8.881784e-16	7.268693e-03
58	5.000	7.500	1.707107e+00	1.707107e+00	0.000000e+00
72	0.000	10.000	-1.000000e+00	-1.000000e+00	0.000000e+00
74	2.500	10.000	-1.707107e+00	-1.707107e+00	0.000000e+00
76	5.000	10.000	0.000000e+00	0.000000e+00	0.000000e+00

Дробление сетки в 4 раза

Параметры дробления

OX: n = 8 q = 1 n = 8 q = 1

OY: n = 8 q = 1 n = 8 q = 1

Абсолютна погрешность: $\|\tilde{U} - U(\frac{h}{4})\| = 0.129679$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	1.000000e+00	1.000000e+00	0.000000e+00
4	2.500	0.000	2.928932e-01	2.928932e-01	0.000000e+00
8	5.000	0.000	2.000000e+00	2.000000e+00	0.000000e+00
12	7.500	0.000	2.928932e-01	2.928932e-01	0.000000e+00
16	10.000	0.000	1.000000e+00	1.000000e+00	0.000000e+00
68	0.000	2.500	-7.071068e-01	-7.071068e-01	0.000000e+00
* 72	2.500	2.500	-1.500722e+00	-1.414214e+00	8.650811e-02
* 76	5.000	2.500	3.031286e-01	2.928932e-01	1.023540e-02
* 80	7.500	2.500	-1.499712e+00	-1.414214e+00	8.549818e-02
84	10.000	2.500	-7.071068e-01	-7.071068e-01	0.000000e+00
136	0.000	5.000	3.061617e-16	3.061617e-16	0.000000e+00
* 140	2.500	5.000	-7.508924e-01	-7.071068e-01	4.378564e-02
144	5.000	5.000	1.000000e+00	1.000000e+00	0.000000e+00
148	7.500	5.000	-7.071068e-01	-7.071068e-01	0.000000e+00
152	10.000	5.000	9.184851e-16	9.184851e-16	0.000000e+00
204	0.000	7.500	7.071068e-01	7.071068e-01	0.000000e+00
* 208	2.500	7.500	-1.009935e-03	-8.881784e-16	1.009935e-03
212	5.000	7.500	1.707107e+00	1.707107e+00	0.000000e+00
272	0.000	10.000	-1.000000e+00	-1.000000e+00	0.000000e+00
276	2.500	10.000	-1.707107e+00	-1.707107e+00	0.000000e+00
280	5.000	10.000	0.000000e+00	0.000000e+00	0.000000e+00

Оценим порядок сходимости:

$$\log_2\left(\frac{\|\widetilde{U}-U(h)\|}{\|\widetilde{U}-U(\frac{h}{2})\|}\right)=\frac{2.61644}{0.553959}=2.23975$$

$$\log_2\left(\frac{\|\widetilde{U}-U(\frac{h}{2})\|}{\|\widetilde{U}-U(\frac{h}{4})\|}\right)=\frac{0.553959}{0.129679}=2.09483$$

Тест №7 (Полином 1-ой степени на неравномерной сетке)

u(x,y) = x+y
λ=1, γ=1
f(x,y) = x+y
Краевые условия 1-ого рода на всех сторонах
Содержание файла CalcArea.txt (Формат описан ниже)
Абсолютна погрешность: $\|\widetilde{U}-U(h)\|$=1.3293e-14

```
3 3
0 0 5 0 10 0
0 5 5 5 10 5
0 10 5 10 10 10
2
1 1 2 1 3
1 2 3 1 2
6
1 1 1 1 1 3
2 1 1 2 3 3
3 1 2 2 2 3
4 1 2 3 2 2
5 1 3 3 1 2
6 1 1 3 1 1
2 4 4 2
2 4 4 2
```

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
1	1.000	0.000	1.000000e+00	1.000000e+00	0.000000e+00
2	5.000	0.000	5.000000e+00	5.000000e+00	0.000000e+00
3	5.333	0.000	5.333333e+00	5.333333e+00	0.000000e+00
4	6.000	0.000	6.000000e+00	6.000000e+00	0.000000e+00
5	7.333	0.000	7.333333e+00	7.333333e+00	0.000000e+00
6	10.000	0.000	1.000000e+01	1.000000e+01	0.000000e+00
7	0.000	1.000	1.000000e+00	1.000000e+00	0.000000e+00
* 8	1.000	1.000	2.000000e+00	2.000000e+00	1.776357e-15
* 9	5.000	1.000	6.000000e+00	6.000000e+00	6.217249e-15
* 10	5.333	1.000	6.333333e+00	6.333333e+00	4.440892e-15
* 11	6.000	1.000	7.000000e+00	7.000000e+00	2.664535e-15
* 12	7.333	1.000	8.333333e+00	8.333333e+00	1.776357e-15
13	10.000	1.000	1.100000e+01	1.100000e+01	0.000000e+00
14	0.000	5.000	5.000000e+00	5.000000e+00	0.000000e+00
* 15	1.000	5.000	6.000000e+00	6.000000e+00	7.105427e-15
16	5.000	5.000	1.000000e+01	1.000000e+01	0.000000e+00
17	5.333	5.000	1.033333e+01	1.033333e+01	0.000000e+00
18	6.000	5.000	1.100000e+01	1.100000e+01	0.000000e+00
19	7.333	5.000	1.233333e+01	1.233333e+01	0.000000e+00
20	10.000	5.000	1.500000e+01	1.500000e+01	0.000000e+00
21	0.000	5.333	5.333333e+00	5.333333e+00	0.000000e+00
* 22	1.000	5.333	6.333333e+00	6.333333e+00	6.217249e-15
23	5.000	5.333	1.033333e+01	1.033333e+01	0.000000e+00
28	0.000	6.000	6.000000e+00	6.000000e+00	0.000000e+00
* 29	1.000	6.000	7.000000e+00	7.000000e+00	3.552714e-15
30	5.000	6.000	1.100000e+01	1.100000e+01	0.000000e+00
35	0.000	7.333	7.333333e+00	7.333333e+00	0.000000e+00
* 36	1.000	7.333	8.333333e+00	8.333333e+00	1.776357e-15
37	5.000	7.333	1.233333e+01	1.233333e+01	0.000000e+00
42	0.000	10.000	1.000000e+01	1.000000e+01	0.000000e+00
43	1.000	10.000	1.100000e+01	1.100000e+01	0.000000e+00
44	5.000	10.000	1.500000e+01	1.500000e+01	0.000000e+00

Тест №9 (Полином 2-ой степени на неравномерной сетке)

$u(x,y) = x^2+y^2$
$\lambda=1$, $\gamma=1$
$f(x,y) = -4+x^2+y^2$
Краевые условия 1-ого рода на всех границах
Содержание файла CalcArea.txt такое же как в Тесте №8
Абсолютна погрешность: $\|\widetilde{U}-U(h)\|$=1.81243e-13

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
1	1.000	0.000	1.000000e+00	1.000000e+00	0.000000e+00
2	5.000	0.000	2.500000e+01	2.500000e+01	0.000000e+00
3	5.333	0.000	2.844444e+01	2.844444e+01	0.000000e+00
4	6.000	0.000	3.600000e+01	3.600000e+01	0.000000e+00
5	7.333	0.000	5.377778e+01	5.377778e+01	0.000000e+00
6	10.000	0.000	1.000000e+02	1.000000e+02	0.000000e+00
7	0.000	1.000	1.000000e+00	1.000000e+00	0.000000e+00
* 8	1.000	1.000	2.000000e+00	2.000000e+00	2.398082e-14
* 9	5.000	1.000	2.600000e+01	2.600000e+01	9.237056e-14
* 10	5.333	1.000	2.944444e+01	2.944444e+01	7.460699e-14
* 11	6.000	1.000	3.700000e+01	3.700000e+01	2.842171e-14
* 12	7.333	1.000	5.477778e+01	5.477778e+01	7.105427e-15
13	10.000	1.000	1.010000e+02	1.010000e+02	0.000000e+00
14	0.000	5.000	2.500000e+01	2.500000e+01	0.000000e+00
* 15	1.000	5.000	2.600000e+01	2.600000e+01	9.947598e-14
16	5.000	5.000	5.000000e+01	5.000000e+01	0.000000e+00
17	5.333	5.000	5.344444e+01	5.344444e+01	0.000000e+00
18	6.000	5.000	6.100000e+01	6.100000e+01	0.000000e+00
19	7.333	5.000	7.877778e+01	7.877778e+01	0.000000e+00
20	10.000	5.000	1.250000e+02	1.250000e+02	0.000000e+00
21	0.000	5.333	2.844444e+01	2.844444e+01	0.000000e+00
* 22	1.000	5.333	2.944444e+01	2.944444e+01	7.815970e-14
23	5.000	5.333	5.344444e+01	5.344444e+01	0.000000e+00
28	0.000	6.000	3.600000e+01	3.600000e+01	0.000000e+00
* 29	1.000	6.000	3.700000e+01	3.700000e+01	3.552714e-14
30	5.000	6.000	6.100000e+01	6.100000e+01	0.000000e+00
35	0.000	7.333	5.377778e+01	5.377778e+01	0.000000e+00
* 36	1.000	7.333	5.477778e+01	5.477778e+01	7.105427e-15
37	5.000	7.333	7.877778e+01	7.877778e+01	0.000000e+00
42	0.000	10.000	1.000000e+02	1.000000e+02	0.000000e+00
43	1.000	10.000	1.010000e+02	1.010000e+02	0.000000e+00
44	5.000	10.000	1.250000e+02	1.250000e+02	0.000000e+00

Тест №10 (Полином третьей степени на не равномерной сетке)

$u(x,y) = x^3+y^3$

$\lambda=1$, $\gamma=1$

$f(x,y) = -6x-6y+x^3+y^3$

Краевые условия 1-ого рода на всех границах

Содержание файла CalcArea.txt такое же как в Тесте №8

Абсолютна погрешность: $\|\widetilde{U} - U(h)\|=11.159$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
1	1.000	0.000	1.000000e+00	1.000000e+00	0.000000e+00
2	5.000	0.000	1.250000e+02	1.250000e+02	0.000000e+00
3	5.333	0.000	1.517037e+02	1.517037e+02	0.000000e+00
4	6.000	0.000	2.160000e+02	2.160000e+02	0.000000e+00
5	7.333	0.000	3.943704e+02	3.943704e+02	0.000000e+00
6	10.000	0.000	1.000000e+03	1.000000e+03	0.000000e+00
7	0.000	1.000	1.000000e+00	1.000000e+00	0.000000e+00
* 8	1.000	1.000	8.079332e+00	2.000000e+00	6.079332e+00
* 9	5.000	1.000	1.267933e+02	1.260000e+02	7.933213e-01
* 10	5.333	1.000	1.548789e+02	1.527037e+02	2.175215e+00
* 11	6.000	1.000	2.208044e+02	2.170000e+02	3.804387e+00
* 12	7.333	1.000	4.002641e+02	3.953704e+02	4.893727e+00
13	10.000	1.000	1.001000e+03	1.001000e+03	0.000000e+00
14	0.000	5.000	1.250000e+02	1.250000e+02	0.000000e+00
* 15	1.000	5.000	1.267933e+02	1.260000e+02	7.933213e-01
16	5.000	5.000	2.500000e+02	2.500000e+02	0.000000e+00
17	5.333	5.000	2.767037e+02	2.767037e+02	0.000000e+00
18	6.000	5.000	3.410000e+02	3.410000e+02	0.000000e+00
19	7.333	5.000	5.193704e+02	5.193704e+02	0.000000e+00
20	10.000	5.000	1.125000e+03	1.125000e+03	0.000000e+00
21	0.000	5.333	1.517037e+02	1.517037e+02	0.000000e+00
* 22	1.000	5.333	1.548789e+02	1.527037e+02	2.175215e+00
23	5.000	5.333	2.767037e+02	2.767037e+02	0.000000e+00
28	0.000	6.000	2.160000e+02	2.160000e+02	0.000000e+00
* 29	1.000	6.000	2.208044e+02	2.170000e+02	3.804387e+00
30	5.000	6.000	3.410000e+02	3.410000e+02	0.000000e+00
35	0.000	7.333	3.943704e+02	3.943704e+02	0.000000e+00
* 36	1.000	7.333	4.002641e+02	3.953704e+02	4.893727e+00
37	5.000	7.333	5.193704e+02	5.193704e+02	0.000000e+00
42	0.000	10.000	1.000000e+03	1.000000e+03	0.000000e+00
43	1.000	10.000	1.001000e+03	1.001000e+03	0.000000e+00
44	5.000	10.000	1.125000e+03	1.125000e+03	0.000000e+00

Произведем дробление

Дробление сетки в 2 раза

Параметры дробления
OX: n = 4 q = 2 n = 8 q = 1.41421
OY: n = 4 q = 2 n = 8 q = 1.41421

Абсолютна погрешность: $\|\widetilde{U} - U(\frac{h}{2})\| = 2.64744$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

	N	X	Y	U	U*	|U* - U|
	0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
	2	1.000	0.000	1.000000e+00	1.000000e+00	0.000000e+00
	4	5.000	0.000	1.250000e+02	1.250000e+02	0.000000e+00
	6	5.333	0.000	1.517040e+02	1.517040e+02	0.000000e+00
	8	6.000	0.000	2.160009e+02	2.160009e+02	0.000000e+00
	10	7.333	0.000	3.943720e+02	3.943720e+02	0.000000e+00
	12	10.000	0.000	1.000000e+03	1.000000e+03	0.000000e+00
	26	0.000	1.000	1.000000e+00	1.000000e+00	0.000000e+00
*	28	1.000	1.000	3.401116e+00	2.000000e+00	1.401116e+00
*	30	5.000	1.000	1.253343e+02	1.260000e+02	6.656821e-01
*	32	5.333	1.000	1.526998e+02	1.527040e+02	4.221545e-03
*	34	6.000	1.000	2.177395e+02	2.170009e+02	7.386540e-01
*	36	7.333	1.000	3.966106e+02	3.953720e+02	1.238606e+00
	38	10.000	1.000	1.001000e+03	1.001000e+03	0.000000e+00
	52	0.000	5.000	1.250000e+02	1.250000e+02	0.000000e+00
*	54	1.000	5.000	1.253343e+02	1.260000e+02	6.656821e-01
	56	5.000	5.000	2.500000e+02	2.500000e+02	0.000000e+00
	58	5.333	5.000	2.767040e+02	2.767040e+02	0.000000e+00
	60	6.000	5.000	3.410009e+02	3.410009e+02	0.000000e+00
	62	7.333	5.000	5.193720e+02	5.193720e+02	0.000000e+00
	64	10.000	5.000	1.125000e+03	1.125000e+03	0.000000e+00
	78	0.000	5.333	1.517040e+02	1.517040e+02	0.000000e+00
*	80	1.000	5.333	1.526998e+02	1.527040e+02	4.221545e-03
	82	5.000	5.333	2.767040e+02	2.767040e+02	0.000000e+00
	104	0.000	6.000	2.160009e+02	2.160009e+02	0.000000e+00
*	106	1.000	6.000	2.177395e+02	2.170009e+02	7.386540e-01
	108	5.000	6.000	3.410009e+02	3.410009e+02	0.000000e+00
	130	0.000	7.333	3.943720e+02	3.943720e+02	0.000000e+00
*	132	1.000	7.333	3.966106e+02	3.953720e+02	1.238606e+00
	134	5.000	7.333	5.193720e+02	5.193720e+02	0.000000e+00
	156	0.000	10.000	1.000000e+03	1.000000e+03	0.000000e+00
	158	1.000	10.000	1.001000e+03	1.001000e+03	0.000000e+00
	160	5.000	10.000	1.125000e+03	1.125000e+03	0.000000e+00

Дробление сетки в 4 раза

Параметры дробления

OX: n = 8 q = 1.41421 n = 16 q = 1.18921

OY: n = 8 q = 1.41421 n = 16 q = 1.18921

Абсолютна погрешность: $\|\widetilde{U} - U(\frac{h}{4})\| = 0.868288$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
4	1.000	0.000	1.000024e+00	1.000024e+00	0.000000e+00
8	5.000	0.000	1.250000e+02	1.250000e+02	0.000000e+00
12	5.333	0.000	1.517031e+02	1.517031e+02	0.000000e+00
16	6.000	0.000	2.159983e+02	2.159983e+02	0.000000e+00
20	7.333	0.000	3.943673e+02	3.943673e+02	0.000000e+00
24	10.000	0.000	1.000000e+03	1.000000e+03	0.000000e+00
100	0.000	1.000	1.000024e+00	1.000024e+00	0.000000e+00
* 104	1.000	1.000	2.347466e+00	2.000048e+00	3.474176e-01
* 108	5.000	1.000	1.255741e+02	1.260000e+02	4.259418e-01
* 112	5.333	1.000	1.525364e+02	1.527031e+02	1.666810e-01
* 116	6.000	1.000	2.171168e+02	2.169983e+02	1.184223e-01
* 120	7.333	1.000	3.956729e+02	3.953673e+02	3.055809e-01
124	10.000	1.000	1.001000e+03	1.001000e+03	0.000000e+00
200	0.000	5.000	1.250000e+02	1.250000e+02	0.000000e+00
* 204	1.000	5.000	1.255741e+02	1.260000e+02	4.259418e-01
208	5.000	5.000	2.500000e+02	2.500000e+02	0.000000e+00
212	5.333	5.000	2.767031e+02	2.767031e+02	0.000000e+00
216	6.000	5.000	3.409983e+02	3.409983e+02	0.000000e+00
220	7.333	5.000	5.193673e+02	5.193673e+02	0.000000e+00
224	10.000	5.000	1.125000e+03	1.125000e+03	0.000000e+00
300	0.000	5.333	1.517031e+02	1.517031e+02	0.000000e+00
* 304	1.000	5.333	1.525364e+02	1.527031e+02	1.666810e-01
308	5.000	5.333	2.767031e+02	2.767031e+02	0.000000e+00
400	0.000	6.000	2.159983e+02	2.159983e+02	0.000000e+00
* 404	1.000	6.000	2.171168e+02	2.169983e+02	1.184223e-01
408	5.000	6.000	3.409983e+02	3.409983e+02	0.000000e+00
500	0.000	7.333	3.943673e+02	3.943673e+02	0.000000e+00
* 504	1.000	7.333	3.956729e+02	3.953673e+02	3.055809e-01
508	5.000	7.333	5.193673e+02	5.193673e+02	0.000000e+00
600	0.000	10.000	1.000000e+03	1.000000e+03	0.000000e+00
604	1.000	10.000	1.001000e+03	1.001000e+03	0.000000e+00
608	5.000	10.000	1.125000e+03	1.125000e+03	0.000000e+00

Дробление сетки в 8 раз

Параметры дробления

OX: n = 16 q = 1.18921 n = 32 q = 1.09051

OY: n = 16 q = 1.18921 n = 32 q = 1.09051

Абсолютна погрешность: $\|\tilde{U} - U(\frac{h}{8})\| = 0.276315$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
8	1.000	0.000	9.999534e-01	9.999534e-01	1.110223e-16
16	5.000	0.000	1.250000e+02	1.250000e+02	1.421085e-14
24	5.333	0.000	1.517026e+02	1.517026e+02	2.842171e-14
32	6.000	0.000	2.159971e+02	2.159971e+02	2.842171e-14
40	7.333	0.000	3.943651e+02	3.943651e+02	5.684342e-14
48	10.000	0.000	1.000000e+03	1.000000e+03	1.136868e-13
392	0.000	1.000	9.999534e-01	9.999534e-01	1.110223e-16
* 400	1.000	1.000	2.086894e+00	1.999907e+00	8.698734e-02
* 408	5.000	1.000	1.258471e+02	1.260000e+02	1.528557e-01
* 416	5.333	1.000	1.526311e+02	1.527026e+02	7.143920e-02
* 424	6.000	1.000	2.170147e+02	2.169971e+02	1.760115e-02
* 432	7.333	1.000	3.954400e+02	3.953651e+02	7.492200e-02
440	10.000	1.000	1.001000e+03	1.001000e+03	1.136868e-13
784	0.000	5.000	1.250000e+02	1.250000e+02	1.421085e-14
* 792	1.000	5.000	1.258471e+02	1.260000e+02	1.528557e-01
800	5.000	5.000	2.500000e+02	2.500000e+02	2.842171e-14
808	5.333	5.000	2.767026e+02	2.767026e+02	5.684342e-14
816	6.000	5.000	3.409971e+02	3.409971e+02	5.684342e-14
824	7.333	5.000	5.193651e+02	5.193651e+02	1.136868e-13
832	10.000	5.000	1.125000e+03	1.125000e+03	2.273737e-13
1176	0.000	5.333	1.517026e+02	1.517026e+02	2.842171e-14
* 1184	1.000	5.333	1.526311e+02	1.527026e+02	7.143920e-02
1192	5.000	5.333	2.767026e+02	2.767026e+02	5.684342e-14
1568	0.000	6.000	2.159971e+02	2.159971e+02	2.842171e-14
* 1576	1.000	6.000	2.170147e+02	2.169971e+02	1.760115e-02
1584	5.000	6.000	3.409971e+02	3.409971e+02	5.684342e-14
1960	0.000	7.333	3.943651e+02	3.943651e+02	5.684342e-14
* 1968	1.000	7.333	3.954400e+02	3.953651e+02	7.492200e-02
1976	5.000	7.333	5.193651e+02	5.193651e+02	1.136868e-13
2352	0.000	10.000	1.000000e+03	1.000000e+03	1.136868e-13
2360	1.000	10.000	1.001000e+03	1.001000e+03	1.136868e-13
2368	5.000	10.000	1.125000e+03	1.125000e+03	2.273737e-13

Дробление сетки в 16 раз

Параметры дробления

OX: n = 32 q = 1.09051 n = 64 q = 1.04427

OY: n = 32 q = 1.09051 n = 64 q = 1.04427

Абсолютна погрешность: $\|\widetilde{U} - U(\frac{h}{16})\| = 0.07604$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
16	1.000	0.000	9.999202e-01	9.999202e-01	1.110223e-16
32	5.000	0.000	1.250000e+02	1.250000e+02	1.421085e-14
48	5.333	0.000	1.517074e+02	1.517074e+02	2.842171e-14
64	6.000	0.000	2.160100e+02	2.160100e+02	2.842171e-14
80	7.333	0.000	3.943887e+02	3.943887e+02	5.684342e-14
96	10.000	0.000	1.000000e+03	1.000000e+03	1.136868e-13
1552	0.000	1.000	9.999202e-01	9.999202e-01	1.110223e-16
* 1568	1.000	1.000	2.021605e+00	1.999840e+00	2.176473e-02
* 1584	5.000	1.000	1.259568e+02	1.259999e+02	4.308763e-02
* 1600	5.333	1.000	1.526863e+02	1.527074e+02	2.107024e-02
* 1616	6.000	1.000	2.170130e+02	2.170099e+02	3.106417e-03
* 1632	7.333	1.000	3.954071e+02	3.953886e+02	1.854836e-02
1648	10.000	1.000	1.001000e+03	1.001000e+03	1.136868e-13
3104	0.000	5.000	1.250000e+02	1.250000e+02	1.421085e-14
* 3120	1.000	5.000	1.259568e+02	1.259999e+02	4.308763e-02
3136	5.000	5.000	2.500000e+02	2.500000e+02	2.842171e-14
3152	5.333	5.000	2.767074e+02	2.767074e+02	5.684342e-14
3168	6.000	5.000	3.410100e+02	3.410100e+02	5.684342e-14
3184	7.333	5.000	5.193887e+02	5.193887e+02	1.136868e-13
3200	10.000	5.000	1.125000e+03	1.125000e+03	2.273737e-13
4656	0.000	5.333	1.517074e+02	1.517074e+02	2.842171e-14
* 4672	1.000	5.333	1.526863e+02	1.527074e+02	2.107024e-02
4688	5.000	5.333	2.767074e+02	2.767074e+02	5.684342e-14
6208	0.000	6.000	2.160100e+02	2.160100e+02	2.842171e-14
* 6224	1.000	6.000	2.170130e+02	2.170099e+02	3.106417e-03
6240	5.000	6.000	3.410100e+02	3.410100e+02	5.684342e-14
7760	0.000	7.333	3.943887e+02	3.943887e+02	5.684342e-14
* 7776	1.000	7.333	3.954071e+02	3.953886e+02	1.854836e-02
7792	5.000	7.333	5.193887e+02	5.193887e+02	1.136868e-13
9312	0.000	10.000	1.000000e+03	1.000000e+03	1.136868e-13
9328	1.000	10.000	1.001000e+03	1.001000e+03	1.136868e-13
9344	5.000	10.000	1.125000e+03	1.125000e+03	2.273737e-13

Оценим порядок сходимости:

$$\log_2\left(\frac{\|\widetilde{U}-U(h)\|}{\|\widetilde{U}-U(\frac{h}{2})\|}\right)=\frac{11.159}{2.64744}=2.07554$$

$$\log_2\left(\frac{\|\widetilde{U}-U(\frac{h}{2})\|}{\|\widetilde{U}-U(\frac{h}{4})\|}\right)=\frac{2.64744}{0.868288}=1.60835$$

$$\log_2\left(\frac{\|\widetilde{U}-U(\frac{h}{4})\|}{\|\widetilde{U}-U(\frac{h}{8})\|}\right)=\frac{0.868288}{0.276315}=1.65186$$

$$\log_2\left(\frac{\|\widetilde{U}-U(\frac{h}{8})\|}{\|\widetilde{U}-U(\frac{h}{16})\|}\right)=\frac{0.276315}{0.07604}=1.86148$$

Тест №11 (Порядок сходимости на не равномерной сетке)

$$u=\sin\frac{\pi x}{2}+\cos\frac{\pi y}{2}, f=\left(\frac{\pi^2}{4}+1\right)*\left(\sin\frac{\pi x}{2}+\cos\frac{\pi y}{2}\right), \gamma=1, \lambda=1$$

Краевые условия 1-ого типа на всех границах

Содержание файла CalcArea.txt

```
3 3
0 0 5 0 10 0
0 5 5 5 10 5
0 10 5 10 10 10
2
1 1 2 1 3
1 2 3 1 2
6
1 1 1 1 1 3
2 1 1 2 3 3
3 1 2 2 2 3
4 1 2 3 2 2
5 1 3 3 1 2
6 1 1 3 1 1
2 4 4 2
2 4 4 2
```

Абсолютна погрешность: $\|\widetilde{U}-U(h)\|=3.35262$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	1.000000e+00	1.000000e+00	6.816769e-14
1	1.000	0.000	2.000000e+00	2.000000e+00	1.363354e-13
2	5.000	0.000	2.000000e+00	2.000000e+00	1.363354e-13
3	5.333	0.000	1.866025e+00	1.866025e+00	1.272316e-13
4	6.000	0.000	1.000000e+00	1.000000e+00	6.816769e-14
5	7.333	0.000	1.339746e-01	1.339746e-01	9.131584e-15
6	10.000	0.000	1.000000e+00	1.000000e+00	6.816769e-14
7	0.000	1.000	6.123234e-17	6.123234e-17	4.178498e-30
* 8	1.000	1.000	2.357339e+00	1.000000e+00	1.357339e+00
* 9	5.000	1.000	2.438181e+00	1.000000e+00	1.438181e+00
* 10	5.333	1.000	1.932046e+00	8.660254e-01	1.066021e+00
* 11	6.000	1.000	6.071770e-01	4.286264e-16	6.071770e-01
* 12	7.333	1.000	-1.361541e+00	-8.660254e-01	4.955158e-01
13	10.000	1.000	6.735557e-16	6.735557e-16	4.595115e-29
14	0.000	5.000	3.061617e-16	3.061617e-16	2.085551e-29
* 15	1.000	5.000	2.034591e+00	1.000000e+00	1.034591e+00
16	5.000	5.000	1.000000e+00	1.000000e+00	6.816769e-14
17	5.333	5.000	8.660254e-01	8.660254e-01	5.895284e-14
18	6.000	5.000	6.735557e-16	6.735557e-16	4.595115e-29
19	7.333	5.000	-8.660254e-01	-8.660254e-01	5.895284e-14
20	10.000	5.000	9.184851e-16	9.184851e-16	6.251723e-29
21	0.000	5.333	-5.000000e-01	-5.000000e-01	3.408385e-14
* 22	1.000	5.333	1.635380e+00	5.000000e-01	1.135380e+00
23	5.000	5.333	5.000000e-01	5.000000e-01	3.408385e-14
28	0.000	6.000	-1.000000e+00	-1.000000e+00	6.816769e-14
* 29	1.000	6.000	1.126746e+00	0.000000e+00	1.126746e+00
30	5.000	6.000	0.000000e+00	0.000000e+00	0.000000e+00
35	0.000	7.333	5.000000e-01	5.000000e-01	3.408385e-14
* 36	1.000	7.333	2.896306e+00	1.500000e+00	1.396306e+00
37	5.000	7.333	1.500000e+00	1.500000e+00	1.021405e-13
42	0.000	10.000	-1.000000e+00	-1.000000e+00	6.816769e-14
43	1.000	10.000	0.000000e+00	0.000000e+00	0.000000e+00
44	5.000	10.000	0.000000e+00	0.000000e+00	0.000000e+00

Дробление сетки в 2 раза

Параметры дробления
OX: n = 4 q = 2 n = 8 q = 1.41421
OY: n = 4 q = 2 n = 8 q = 1.41421

Абсолютна погрешность: $\|\widetilde{U} - U(\frac{h}{2})\| = 1.00765$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	1.000000e+00	1.000000e+00	0.000000e+00
2	1.000	0.000	2.000000e+00	2.000000e+00	0.000000e+00
4	5.000	0.000	2.000000e+00	2.000000e+00	0.000000e+00
6	5.333	0.000	1.866022e+00	1.866022e+00	0.000000e+00
8	6.000	0.000	9.999873e-01	9.999873e-01	0.000000e+00
10	7.333	0.000	1.339823e-01	1.339823e-01	0.000000e+00
12	10.000	0.000	1.000000e+00	1.000000e+00	0.000000e+00
26	0.000	1.000	6.123234e-17	6.123234e-17	0.000000e+00
* 28	1.000	1.000	1.163844e+00	1.000000e+00	1.638436e-01
* 30	5.000	1.000	1.683875e+00	1.000000e+00	6.838745e-01
* 32	5.333	1.000	1.311959e+00	8.660224e-01	4.459366e-01
* 34	6.000	1.000	1.813757e-01	-1.266182e-05	1.813884e-01
* 36	7.333	1.000	-9.865247e-01	-8.660177e-01	1.205070e-01
38	10.000	1.000	6.735557e-16	6.735557e-16	0.000000e+00
52	0.000	5.000	3.061617e-16	3.061617e-16	0.000000e+00
* 54	1.000	5.000	5.633274e-01	1.000000e+00	4.366726e-01
56	5.000	5.000	1.000000e+00	1.000000e+00	0.000000e+00
58	5.333	5.000	8.660224e-01	8.660224e-01	0.000000e+00
60	6.000	5.000	-1.266182e-05	-1.266182e-05	0.000000e+00
62	7.333	5.000	-8.660177e-01	-8.660177e-01	0.000000e+00
64	10.000	5.000	9.184851e-16	9.184851e-16	0.000000e+00
78	0.000	5.333	-5.000052e-01	-5.000052e-01	0.000000e+00
* 80	1.000	5.333	2.603891e-01	4.999948e-01	2.396057e-01
82	5.000	5.333	4.999948e-01	4.999948e-01	0.000000e+00
104	0.000	6.000	-1.000000e+00	-1.000000e+00	0.000000e+00
* 106	1.000	6.000	-5.896858e-02	8.016088e-11	5.896858e-02
108	5.000	6.000	8.016088e-11	8.016088e-11	0.000000e+00
130	0.000	7.333	5.000134e-01	5.000134e-01	0.000000e+00
* 132	1.000	7.333	1.651612e+00	1.500013e+00	1.515983e-01
134	5.000	7.333	1.500013e+00	1.500013e+00	0.000000e+00
156	0.000	10.000	-1.000000e+00	-1.000000e+00	0.000000e+00
158	1.000	10.000	0.000000e+00	0.000000e+00	0.000000e+00
160	5.000	10.000	0.000000e+00	0.000000e+00	0.000000e+00

Дробление сетки в 4 раза

Параметры дробления

OX: n = 8 q = 1.41421 n = 16 q = 1.18921

OY: n = 8 q = 1.41421 n = 16 q = 1.18921

Абсолютна погрешность: $\|\tilde{U} - U(\frac{h}{4})\| = 0.462115$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	1.000000e+00	1.000000e+00	0.000000e+00
4	1.000	0.000	2.000000e+00	2.000000e+00	0.000000e+00
8	5.000	0.000	2.000000e+00	2.000000e+00	0.000000e+00
12	5.333	0.000	1.866031e+00	1.866031e+00	0.000000e+00
16	6.000	0.000	1.000024e+00	1.000024e+00	0.000000e+00
20	7.333	0.000	1.339597e-01	1.339597e-01	0.000000e+00
24	10.000	0.000	1.000000e+00	1.000000e+00	0.000000e+00
100	0.000	1.000	-1.266182e-05	-1.266182e-05	0.000000e+00
* 104	1.000	1.000	1.052787e+00	9.999873e-01	5.279979e-02
* 108	5.000	1.000	1.196954e+00	9.999873e-01	1.969667e-01
* 112	5.333	1.000	9.986284e-01	8.660185e-01	1.326099e-01
* 116	6.000	1.000	6.166280e-02	1.172666e-05	6.165108e-02
* 120	7.333	1.000	-8.817725e-01	-8.660530e-01	1.571949e-02
124	10.000	1.000	-1.266182e-05	-1.266182e-05	0.000000e+00
200	0.000	5.000	3.061617e-16	3.061617e-16	0.000000e+00
* 204	1.000	5.000	6.752326e-01	1.000000e+00	3.247674e-01
208	5.000	5.000	1.000000e+00	1.000000e+00	0.000000e+00
212	5.333	5.000	8.660312e-01	8.660312e-01	0.000000e+00
216	6.000	5.000	2.438848e-05	2.438848e-05	0.000000e+00
220	7.333	5.000	-8.660403e-01	-8.660403e-01	0.000000e+00
224	10.000	5.000	9.184851e-16	9.184851e-16	0.000000e+00
300	0.000	5.333	-4.999900e-01	-4.999900e-01	0.000000e+00
* 304	1.000	5.333	3.027862e-01	5.000100e-01	1.972238e-01
308	5.000	5.333	5.000100e-01	5.000100e-01	0.000000e+00
400	0.000	6.000	-1.000000e+00	-1.000000e+00	0.000000e+00
* 404	1.000	6.000	-6.906862e-02	2.172380e-10	6.906862e-02
408	5.000	6.000	2.973989e-10	2.973989e-10	0.000000e+00
500	0.000	7.333	4.999742e-01	4.999742e-01	0.000000e+00
* 504	1.000	7.333	1.534493e+00	1.499974e+00	3.451927e-02
508	5.000	7.333	1.499974e+00	1.499974e+00	0.000000e+00
600	0.000	10.000	-1.000000e+00	-1.000000e+00	0.000000e+00
604	1.000	10.000	-8.016088e-11	-8.016088e-11	0.000000e+00
608	5.000	10.000	0.000000e+00	0.000000e+00	0.000000e+00

Дробление сетки в 8 раз

Параметры дробления

OX: n = 16 q = 1.18921 n = 32 q = 1.09051

OY: n = 16 q = 1.18921 n = 32 q = 1.09051

Абсолютна погрешность: $\|\tilde{U} - U(\frac{h}{8})\| = 0.14425$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	1.000000e+00	1.000000e+00	0.000000e+00
8	1.000	0.000	2.000000e+00	2.000000e+00	2.220446e-16
16	5.000	0.000	2.000000e+00	2.000000e+00	0.000000e+00
24	5.333	0.000	1.866035e+00	1.866035e+00	2.220446e-16
32	6.000	0.000	1.000042e+00	1.000042e+00	2.220446e-16
40	7.333	0.000	1.339491e-01	1.339491e-01	2.775558e-17
48	10.000	0.000	1.000000e+00	1.000000e+00	2.220446e-16
392	0.000	1.000	2.438848e-05	2.438848e-05	3.388132e-21
* 400	1.000	1.000	1.013544e+00	1.000024e+00	1.351954e-02
* 408	5.000	1.000	1.037644e+00	1.000024e+00	3.761927e-02
* 416	5.333	1.000	8.916251e-01	8.660597e-01	2.556546e-02
* 424	6.000	1.000	1.219306e-02	6.619191e-05	1.212687e-02
* 432	7.333	1.000	-8.704860e-01	-8.660266e-01	4.459445e-03
440	10.000	1.000	2.438848e-05	2.438848e-05	3.388132e-21
784	0.000	5.000	3.061617e-16	3.061617e-16	4.930381e-32
* 792	1.000	5.000	8.867016e-01	1.000000e+00	1.132984e-01
800	5.000	5.000	1.000000e+00	1.000000e+00	2.220446e-16
808	5.333	5.000	8.660353e-01	8.660353e-01	1.110223e-16
816	6.000	5.000	4.180343e-05	4.180343e-05	6.776264e-21
824	7.333	5.000	-8.660509e-01	-8.660509e-01	1.110223e-16
832	10.000	5.000	9.184851e-16	9.184851e-16	1.972152e-31
1176	0.000	5.333	-4.999829e-01	-4.999829e-01	5.551115e-17
* 1184	1.000	5.333	4.303226e-01	5.000171e-01	6.969446e-02
1192	5.000	5.333	5.000171e-01	5.000171e-01	1.110223e-16
1568	0.000	6.000	-1.000000e+00	-1.000000e+00	1.110223e-16
* 1576	1.000	6.000	-2.518752e-02	5.763646e-10	2.518752e-02
1584	5.000	6.000	8.737635e-10	8.737635e-10	1.033976e-25
1960	0.000	7.333	4.999558e-01	4.999558e-01	5.551115e-17
* 1968	1.000	7.333	1.507785e+00	1.499956e+00	7.829609e-03
1976	5.000	7.333	1.499956e+00	1.499956e+00	2.220446e-16
2352	0.000	10.000	-1.000000e+00	-1.000000e+00	0.000000e+00
2360	1.000	10.000	-2.973989e-10	-2.973989e-10	5.169879e-26
2368	5.000	10.000	0.000000e+00	0.000000e+00	0.000000e+00

Дробление сетки в 16 раз

Параметры дробления

OX: n = 32 q = 1.09051 n = 64 q = 1.04427

OY: n = 32 q = 1.09051 n = 64 q = 1.04427

Абсолютна погрешность: $\|\widetilde{U} - U(\frac{h}{16})\| = 0.0387903$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	1.000000e+00	1.000000e+00	0.000000e+00
16	1.000	0.000	2.000000e+00	2.000000e+00	2.220446e-16
32	5.000	0.000	2.000000e+00	2.000000e+00	0.000000e+00
48	5.333	0.000	1.865991e+00	1.865991e+00	2.220446e-16
64	6.000	0.000	9.998543e-01	9.998543e-01	1.110223e-16
80	7.333	0.000	1.340636e-01	1.340636e-01	2.775558e-17
96	10.000	0.000	1.000000e+00	1.000000e+00	2.220446e-16
1552	0.000	1.000	4.180343e-05	4.180343e-05	6.776264e-21
* 1568	1.000	1.000	1.003440e+00	1.000042e+00	3.398500e-03
* 1584	5.000	1.000	1.007942e+00	1.000042e+00	7.899895e-03
* 1600	5.333	1.000	8.714455e-01	8.660328e-01	5.412715e-03
* 1616	6.000	1.000	2.508059e-03	-1.038538e-04	2.611913e-03
* 1632	7.333	1.000	-8.670902e-01	-8.658946e-01	1.195602e-03
1648	10.000	1.000	4.180343e-05	4.180343e-05	6.776264e-21
3104	0.000	5.000	3.061617e-16	3.061617e-16	4.930381e-32
* 3120	1.000	5.000	9.688653e-01	1.000000e+00	3.113469e-02
3136	5.000	5.000	1.000000e+00	1.000000e+00	2.220446e-16
3152	5.333	5.000	8.659910e-01	8.659910e-01	1.110223e-16
3168	6.000	5.000	-1.456572e-04	-1.456572e-04	2.710505e-20
3184	7.333	5.000	-8.659364e-01	-8.659364e-01	1.110223e-16
3200	10.000	5.000	9.184851e-16	9.184851e-16	1.972152e-31
4656	0.000	5.333	-5.000596e-01	-5.000596e-01	1.110223e-16
* 4672	1.000	5.333	4.806750e-01	4.999404e-01	1.926547e-02
4688	5.000	5.333	4.999404e-01	4.999404e-01	5.551115e-17
6208	0.000	6.000	-1.000000e+00	-1.000000e+00	1.110223e-16
* 6224	1.000	6.000	-7.009549e-03	9.734246e-09	7.009559e-03
6240	5.000	6.000	1.060801e-08	1.060801e-08	1.654361e-24
7760	0.000	7.333	5.001542e-01	5.001542e-01	1.110223e-16
* 7776	1.000	7.333	1.502029e+00	1.500154e+00	1.874864e-03
7792	5.000	7.333	1.500154e+00	1.500154e+00	2.220446e-16
9312	0.000	10.000	-1.000000e+00	-1.000000e+00	0.000000e+00
9328	1.000	10.000	-8.737635e-10	-8.737635e-10	1.033976e-25
9344	5.000	10.000	0.000000e+00	0.000000e+00	0.000000e+00

Оценим порядок сходимости:

$$\log_2\left(\frac{\|\widetilde{U}-U(h)\|}{\|\widetilde{U}-U(\frac{h}{2})\|}\right)=\frac{3.35262}{1.00765}=1.73429$$

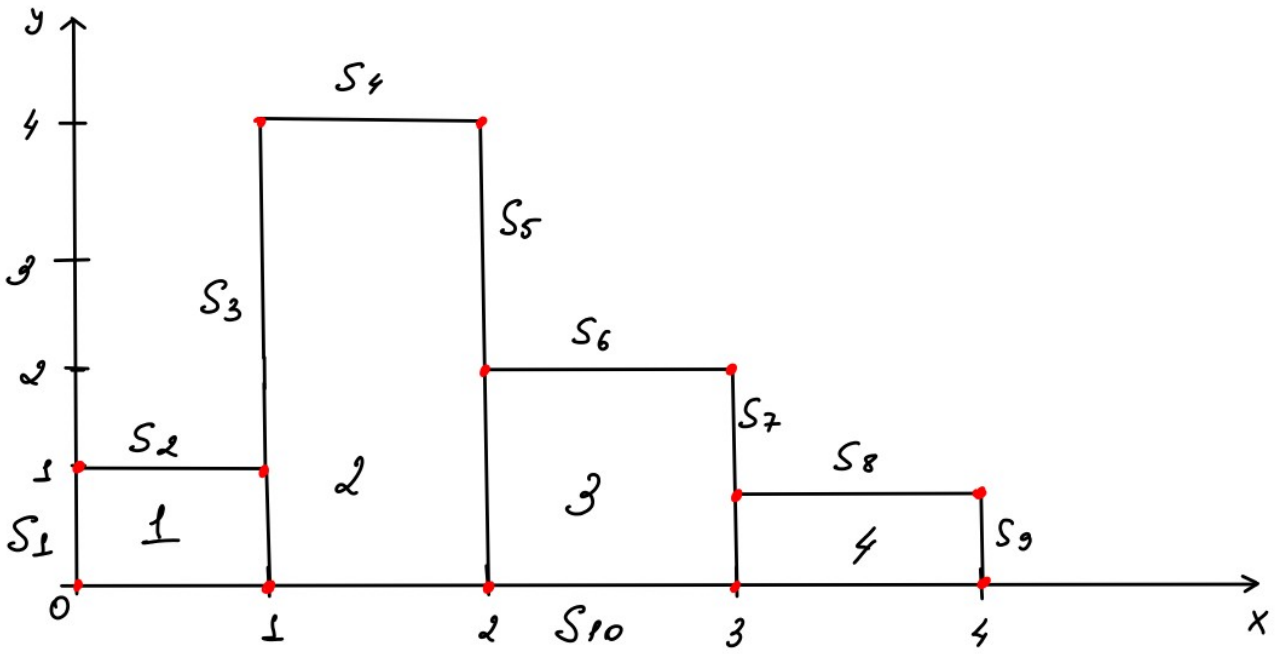
$$\log_2\left(\frac{\|\widetilde{U}-U(\frac{h}{2})\|}{\|\widetilde{U}-U(\frac{h}{4})\|}\right)=\frac{1.00765}{0.462115}=1.12467$$

$$\log_2\left(\frac{\|\widetilde{U}-U(\frac{h}{4})\|}{\|\widetilde{U}-U(\frac{h}{8})\|}\right)=\frac{0.462115}{0.14425}=1.67968$$

$$\log_2\left(\frac{\|\widetilde{U}-U(\frac{h}{8})\|}{\|\widetilde{U}-U(\frac{h}{16})\|}\right)=\frac{0.14425}{0.0387903}=1.8948$$

Тестирование программы для решения на произвольной области

Расчетная область (это не равномерная сетка по построению)



Тест №12 (Полином 1-ой степени**)**

u(x,y) = x+y

λ=1, γ=1

f(x,y) = x+y

Краевые условия 1-ого рода на всех сторонах

Содержание файла CalcArea.txt (Формат описан ниже)

```
5 4
0 0 1 0 2 0 3 0 4 0
0 1 1 1 2 1 3 1 4 1
0 2 1 2 2 2 3 2 4 2
0 4 1 4 2 4 3 4 4 4
4
1 1 2 1 2
1 2 3 1 4
1 3 4 1 3
1 4 5 1 2
10
1 1 1 1 1 2
2 1 1 2 2 2
3 1 2 2 2 4
4 1 2 3 4 4
5 1 3 3 3 4
6 1 3 4 3 3
```

Продолжение файла расчетной обл.

```
7 1 4 4 2 3
8 1 4 5 2 2
9 1 5 5 1 2
10 1 1 5 1 1
2 1 2 1 2 1 2 1
2 1 2 1 2 1
```

Абсолютна погрешность: $\|\widetilde{U} - U(h)\| = 2.01183e\text{-}14$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
1	0.500	0.000	5.000000e-01	5.000000e-01	0.000000e+00
2	1.000	0.000	1.000000e+00	1.000000e+00	0.000000e+00
3	1.500	0.000	1.500000e+00	1.500000e+00	0.000000e+00
4	2.000	0.000	2.000000e+00	2.000000e+00	0.000000e+00
5	2.500	0.000	2.500000e+00	2.500000e+00	0.000000e+00
6	3.000	0.000	3.000000e+00	3.000000e+00	0.000000e+00
7	3.500	0.000	3.500000e+00	3.500000e+00	0.000000e+00
8	4.000	0.000	4.000000e+00	4.000000e+00	0.000000e+00
9	0.000	0.500	5.000000e-01	5.000000e-01	0.000000e+00
* 10	0.500	0.500	1.000000e+00	1.000000e+00	2.775558e-15
* 11	1.000	0.500	1.500000e+00	1.500000e+00	5.551115e-15
* 12	1.500	0.500	2.000000e+00	2.000000e+00	9.769963e-15
* 13	2.000	0.500	2.500000e+00	2.500000e+00	8.437695e-15
* 14	2.500	0.500	3.000000e+00	3.000000e+00	4.440892e-15
* 15	3.000	0.500	3.500000e+00	3.500000e+00	1.332268e-15
* 16	3.500	0.500	4.000000e+00	4.000000e+00	0.000000e+00
17	4.000	0.500	4.500000e+00	4.500000e+00	0.000000e+00
18	0.000	1.000	1.000000e+00	1.000000e+00	0.000000e+00
19	0.500	1.000	1.500000e+00	1.500000e+00	0.000000e+00
20	1.000	1.000	2.000000e+00	2.000000e+00	0.000000e+00
* 21	1.500	1.000	2.500000e+00	2.500000e+00	8.437695e-15
* 22	2.000	1.000	3.000000e+00	3.000000e+00	7.549517e-15
* 23	2.500	1.000	3.500000e+00	3.500000e+00	3.552714e-15
24	3.000	1.000	4.000000e+00	4.000000e+00	0.000000e+00
25	3.500	1.000	4.500000e+00	4.500000e+00	0.000000e+00
26	4.000	1.000	5.000000e+00	5.000000e+00	0.000000e+00
29	1.000	1.500	2.500000e+00	2.500000e+00	0.000000e+00
* 30	1.500	1.500	3.000000e+00	3.000000e+00	4.440892e-15
* 31	2.000	1.500	3.500000e+00	3.500000e+00	3.552714e-15
* 32	2.500	1.500	4.000000e+00	4.000000e+00	1.776357e-15
33	3.000	1.500	4.500000e+00	4.500000e+00	0.000000e+00
38	1.000	2.000	3.000000e+00	3.000000e+00	0.000000e+00
* 39	1.500	2.000	3.500000e+00	3.500000e+00	1.332268e-15
40	2.000	2.000	4.000000e+00	4.000000e+00	0.000000e+00
41	2.500	2.000	4.500000e+00	4.500000e+00	0.000000e+00
42	3.000	2.000	5.000000e+00	5.000000e+00	0.000000e+00
47	1.000	3.000	4.000000e+00	4.000000e+00	0.000000e+00
* 48	1.500	3.000	4.500000e+00	4.500000e+00	0.000000e+00
49	2.000	3.000	5.000000e+00	5.000000e+00	0.000000e+00
56	1.000	4.000	5.000000e+00	5.000000e+00	0.000000e+00
57	1.500	4.000	5.500000e+00	5.500000e+00	0.000000e+00
58	2.000	4.000	6.000000e+00	6.000000e+00	0.000000e+00

Тест №13 (Полином третьей степени)

$u(x,y) = x^3+y^3$

$\lambda=1$, $\gamma=1$

$f(x,y) = -6x-6y+x^3+y^3$

Краевые условия 1-ого рода на всех границах

Содержание файла CalcArea.txt такое же как в Тесте №12

Абсолютна погрешность: $\|\widetilde{U}-U(h)\|=0.0863457$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

	N	X	Y	U	U*	|U* - U|
	0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
	1	0.500	0.000	1.250000e-01	1.250000e-01	0.000000e+00
	2	1.000	0.000	1.000000e+00	1.000000e+00	0.000000e+00
	3	1.500	0.000	3.375000e+00	3.375000e+00	0.000000e+00
	4	2.000	0.000	8.000000e+00	8.000000e+00	0.000000e+00
	5	2.500	0.000	1.562500e+01	1.562500e+01	0.000000e+00
	6	3.000	0.000	2.700000e+01	2.700000e+01	0.000000e+00
	7	3.500	0.000	4.287500e+01	4.287500e+01	0.000000e+00
	8	4.000	0.000	6.400000e+01	6.400000e+01	0.000000e+00
	9	0.000	0.500	1.250000e-01	1.250000e-01	0.000000e+00
*	10	0.500	0.500	2.501213e-01	2.500000e-01	1.213268e-04
*	11	1.000	0.500	1.125516e+00	1.125000e+00	5.156389e-04
*	12	1.500	0.500	3.502070e+00	3.500000e+00	2.070138e-03
*	13	2.000	0.500	8.126585e+00	8.125000e+00	1.585187e-03
*	14	2.500	0.500	1.575078e+01	1.575000e+01	7.843882e-04
*	15	3.000	0.500	2.712520e+01	2.712500e+01	1.953787e-04
*	16	3.500	0.500	4.300005e+01	4.300000e+01	4.597147e-05
	17	4.000	0.500	6.412500e+01	6.412500e+01	0.000000e+00
	18	0.000	1.000	1.000000e+00	1.000000e+00	0.000000e+00
	19	0.500	1.000	1.125000e+00	1.125000e+00	0.000000e+00
	20	1.000	1.000	2.000000e+00	2.000000e+00	0.000000e+00
*	21	1.500	1.000	4.381697e+00	4.375000e+00	6.697263e-03
*	22	2.000	1.000	9.003883e+00	9.000000e+00	3.882516e-03
*	23	2.500	1.000	1.662655e+01	1.662500e+01	1.553085e-03
	24	3.000	1.000	2.800000e+01	2.800000e+01	0.000000e+00
	25	3.500	1.000	4.387500e+01	4.387500e+01	0.000000e+00
	26	4.000	1.000	6.500000e+01	6.500000e+01	0.000000e+00
	29	1.000	1.500	4.375000e+00	4.375000e+00	0.000000e+00
*	30	1.500	1.500	6.772511e+00	6.750000e+00	2.251071e-02
*	31	2.000	1.500	1.138167e+01	1.137500e+01	6.665161e-03
*	32	2.500	1.500	1.900193e+01	1.900000e+01	1.933705e-03
	33	3.000	1.500	3.037500e+01	3.037500e+01	0.000000e+00
	38	1.000	2.000	9.000000e+00	9.000000e+00	0.000000e+00
*	39	1.500	2.000	1.145731e+01	1.137500e+01	8.230810e-02
	40	2.000	2.000	1.600000e+01	1.600000e+01	0.000000e+00
	41	2.500	2.000	2.362500e+01	2.362500e+01	0.000000e+00
	42	3.000	2.000	3.500000e+01	3.500000e+01	0.000000e+00
	47	1.000	3.000	2.800000e+01	2.800000e+01	0.000000e+00
*	48	1.500	3.000	3.038248e+01	3.037500e+01	7.482555e-03
	49	2.000	3.000	3.500000e+01	3.500000e+01	0.000000e+00
	56	1.000	4.000	6.500000e+01	6.500000e+01	0.000000e+00
	57	1.500	4.000	6.737500e+01	6.737500e+01	0.000000e+00
	58	2.000	4.000	7.200000e+01	7.200000e+01	0.000000e+00

Дробления сетки

Дробление сетки на 2

Параметры дробления
OX: n = 4 q = 1 n = 4 q = 1 n = 4 q = 1 n = 4 q = 1
OY: n = 4 q = 1 n = 4 q = 1 n = 4 q = 1 n = 4 q = 1

Абсолютна погрешность: $\|\widetilde{U} - U(\frac{h}{2})\| = 0.0301859$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
2	0.500	0.000	1.250000e-01	1.250000e-01	0.000000e+00
4	1.000	0.000	1.000000e+00	1.000000e+00	0.000000e+00
6	1.500	0.000	3.375000e+00	3.375000e+00	0.000000e+00
8	2.000	0.000	8.000000e+00	8.000000e+00	0.000000e+00
10	2.500	0.000	1.562500e+01	1.562500e+01	0.000000e+00
12	3.000	0.000	2.700000e+01	2.700000e+01	0.000000e+00
14	3.500	0.000	4.287500e+01	4.287500e+01	0.000000e+00
16	4.000	0.000	6.400000e+01	6.400000e+01	0.000000e+00
34	0.000	0.500	1.250000e-01	1.250000e-01	0.000000e+00
* 36	0.500	0.500	2.500555e-01	2.500000e-01	5.553740e-05
* 38	1.000	0.500	1.125259e+00	1.125000e+00	2.585167e-04
* 40	1.500	0.500	3.500683e+00	3.500000e+00	6.829891e-04
* 42	2.000	0.500	8.125685e+00	8.125000e+00	6.845416e-04
* 44	2.500	0.500	1.575039e+01	1.575000e+01	3.865659e-04
* 46	3.000	0.500	2.712512e+01	2.712500e+01	1.189356e-04
* 48	3.500	0.500	4.300003e+01	4.300000e+01	2.503923e-05
50	4.000	0.500	6.412500e+01	6.412500e+01	0.000000e+00
68	0.000	1.000	1.000000e+00	1.000000e+00	0.000000e+00
70	0.500	1.000	1.125000e+00	1.125000e+00	0.000000e+00
72	1.000	1.000	2.000000e+00	2.000000e+00	0.000000e+00
* 74	1.500	1.000	4.377146e+00	4.375000e+00	2.146063e-03
* 76	2.000	1.000	9.001853e+00	9.000000e+00	1.852845e-03
* 78	2.500	1.000	1.662582e+01	1.662500e+01	8.166602e-04
80	3.000	1.000	2.800000e+01	2.800000e+01	0.000000e+00
82	3.500	1.000	4.387500e+01	4.387500e+01	0.000000e+00
84	4.000	1.000	6.500000e+01	6.500000e+01	0.000000e+00
106	1.000	1.500	4.375000e+00	4.375000e+00	0.000000e+00
* 108	1.500	1.500	6.757324e+00	6.750000e+00	7.323846e-03
* 110	2.000	1.500	1.137888e+01	1.137500e+01	3.882590e-03
* 112	2.500	1.500	1.900106e+01	1.900000e+01	1.061537e-03
114	3.000	1.500	3.037500e+01	3.037500e+01	0.000000e+00
140	1.000	2.000	9.000000e+00	9.000000e+00	0.000000e+00
* 142	1.500	2.000	1.140379e+01	1.137500e+01	2.879180e-02
144	2.000	2.000	1.600000e+01	1.600000e+01	0.000000e+00
146	2.500	2.000	2.362500e+01	2.362500e+01	0.000000e+00
148	3.000	2.000	3.500000e+01	3.500000e+01	0.000000e+00
174	1.000	3.000	2.800000e+01	2.800000e+01	0.000000e+00
* 176	1.500	3.000	3.037658e+01	3.037500e+01	1.582545e-03
178	2.000	3.000	3.500000e+01	3.500000e+01	0.000000e+00
208	1.000	4.000	6.500000e+01	6.500000e+01	0.000000e+00
210	1.500	4.000	6.737500e+01	6.737500e+01	0.000000e+00
212	2.000	4.000	7.200000e+01	7.200000e+01	0.000000e+00

Дробление сетки в 4 раза

Параметры дробления

OX: n = 8 q = 1 n = 8 q = 1 n = 8 q = 1 n = 8 q = 1

OY: n = 8 q = 1 n = 8 q = 1 n = 8 q = 1

Абсолютна погрешность: $\|\widetilde{U} - U(\frac{h}{4})\| = 0.00862528$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
4	0.500	0.000	1.250000e-01	1.250000e-01	0.000000e+00
8	1.000	0.000	1.000000e+00	1.000000e+00	0.000000e+00
12	1.500	0.000	3.375000e+00	3.375000e+00	0.000000e+00
16	2.000	0.000	8.000000e+00	8.000000e+00	0.000000e+00
20	2.500	0.000	1.562500e+01	1.562500e+01	0.000000e+00
24	3.000	0.000	2.700000e+01	2.700000e+01	0.000000e+00
28	3.500	0.000	4.287500e+01	4.287500e+01	0.000000e+00
32	4.000	0.000	6.400000e+01	6.400000e+01	0.000000e+00
132	0.000	0.500	1.250000e-01	1.250000e-01	0.000000e+00
* 136	0.500	0.500	2.500175e-01	2.500000e-01	1.746119e-05
* 140	1.000	0.500	1.125082e+00	1.125000e+00	8.244830e-05
* 144	1.500	0.500	3.500199e+00	3.500000e+00	1.985027e-04
* 148	2.000	0.500	8.125208e+00	8.125000e+00	2.078786e-04
* 152	2.500	0.500	1.575012e+01	1.575000e+01	1.231962e-04
* 156	3.000	0.500	2.712504e+01	2.712500e+01	4.089081e-05
* 160	3.500	0.500	4.300001e+01	4.300000e+01	8.401104e-06
164	4.000	0.500	6.412500e+01	6.412500e+01	0.000000e+00
264	0.000	1.000	1.000000e+00	1.000000e+00	0.000000e+00
268	0.500	1.000	1.125000e+00	1.125000e+00	0.000000e+00
272	1.000	1.000	2.000000e+00	2.000000e+00	0.000000e+00
* 276	1.500	1.000	4.375615e+00	4.375000e+00	6.152954e-04
* 280	2.000	1.000	9.000574e+00	9.000000e+00	5.742025e-04
* 284	2.500	1.000	1.662527e+01	1.662500e+01	2.667652e-04
288	3.000	1.000	2.800000e+01	2.800000e+01	0.000000e+00
292	3.500	1.000	4.387500e+01	4.387500e+01	0.000000e+00
296	4.000	1.000	6.500000e+01	6.500000e+01	0.000000e+00
404	1.000	1.500	4.375000e+00	4.375000e+00	0.000000e+00
* 408	1.500	1.500	6.752083e+00	6.750000e+00	2.082644e-03
* 412	2.000	1.500	1.137629e+01	1.137500e+01	1.294069e-03
* 416	2.500	1.500	1.900035e+01	1.900000e+01	3.541183e-04
420	3.000	1.500	3.037500e+01	3.037500e+01	0.000000e+00
536	1.000	2.000	9.000000e+00	9.000000e+00	0.000000e+00
* 540	1.500	2.000	1.138320e+01	1.137500e+01	8.200031e-03
544	2.000	2.000	1.600000e+01	1.600000e+01	0.000000e+00
548	2.500	2.000	2.362500e+01	2.362500e+01	0.000000e+00
552	3.000	2.000	3.500000e+01	3.500000e+01	0.000000e+00
668	1.000	3.000	2.800000e+01	2.800000e+01	0.000000e+00
* 672	1.500	3.000	3.037536e+01	3.037500e+01	3.624508e-04
676	2.000	3.000	3.500000e+01	3.500000e+01	0.000000e+00
800	1.000	4.000	6.500000e+01	6.500000e+01	0.000000e+00
804	1.500	4.000	6.737500e+01	6.737500e+01	0.000000e+00
808	2.000	4.000	7.200000e+01	7.200000e+01	0.000000e+00

Дробление сетки в 8 раза

Параметры дробления

OX: n = 16 q = 1 n = 16 q = 1 n = 16 q = 1 n = 16 q = 1

OY: n = 16 q = 1 n = 16 q = 1 n = 16 q = 1

Абсолютна погрешность: $\|\widetilde{U} - U(\frac{h}{8})\| = 0.00223976$

Расчетная таблица. Символом * отмечены внутренние узлы сетки

N	X	Y	U	U*	|U* - U|
0	0.000	0.000	0.000000e+00	0.000000e+00	0.000000e+00
8	0.500	0.000	1.250000e-01	1.250000e-01	0.000000e+00
16	1.000	0.000	1.000000e+00	1.000000e+00	0.000000e+00
24	1.500	0.000	3.375000e+00	3.375000e+00	0.000000e+00
32	2.000	0.000	8.000000e+00	8.000000e+00	0.000000e+00
40	2.500	0.000	1.562500e+01	1.562500e+01	0.000000e+00
48	3.000	0.000	2.700000e+01	2.700000e+01	0.000000e+00
56	3.500	0.000	4.287500e+01	4.287500e+01	0.000000e+00
64	4.000	0.000	6.400000e+01	6.400000e+01	0.000000e+00
520	0.000	0.500	1.250000e-01	1.250000e-01	0.000000e+00
* 528	0.500	0.500	2.500048e-01	2.500000e-01	4.755272e-06
* 536	1.000	0.500	1.125022e+00	1.125000e+00	2.240547e-05
* 544	1.500	0.500	3.500053e+00	3.500000e+00	5.274994e-05
* 552	2.000	0.500	8.125056e+00	8.125000e+00	5.583579e-05
* 560	2.500	0.500	1.575003e+01	1.575000e+01	3.359573e-05
* 568	3.000	0.500	2.712501e+01	2.712500e+01	1.140752e-05
* 576	3.500	0.500	4.300000e+01	4.300000e+01	2.340266e-06
584	4.000	0.500	6.412500e+01	6.412500e+01	0.000000e+00
1040	0.000	1.000	1.000000e+00	1.000000e+00	0.000000e+00
1048	0.500	1.000	1.125000e+00	1.125000e+00	0.000000e+00
1056	1.000	1.000	2.000000e+00	2.000000e+00	0.000000e+00
* 1064	1.500	1.000	4.375163e+00	4.375000e+00	1.625171e-04
* 1072	2.000	1.000	9.000155e+00	9.000000e+00	1.549167e-04
* 1080	2.500	1.000	1.662507e+01	1.662500e+01	7.334991e-05
1088	3.000	1.000	2.800000e+01	2.800000e+01	0.000000e+00
1096	3.500	1.000	4.387500e+01	4.387500e+01	0.000000e+00
1104	4.000	1.000	6.500000e+01	6.500000e+01	0.000000e+00
1576	1.000	1.500	4.375000e+00	4.375000e+00	0.000000e+00
* 1584	1.500	1.500	6.750546e+00	6.750000e+00	5.463010e-04
* 1592	2.000	1.500	1.137536e+01	1.137500e+01	3.558981e-04
* 1600	2.500	1.500	1.900010e+01	1.900000e+01	9.844485e-05
1608	3.000	1.500	3.037500e+01	3.037500e+01	0.000000e+00
2096	1.000	2.000	9.000000e+00	9.000000e+00	0.000000e+00
* 2104	1.500	2.000	1.137712e+01	1.137500e+01	2.123783e-03
2112	2.000	2.000	1.600000e+01	1.600000e+01	0.000000e+00
2120	2.500	2.000	2.362500e+01	2.362500e+01	0.000000e+00
2128	3.000	2.000	3.500000e+01	3.500000e+01	0.000000e+00
2616	1.000	3.000	2.800000e+01	2.800000e+01	0.000000e+00
* 2624	1.500	3.000	3.037509e+01	3.037500e+01	8.832446e-05
2632	2.000	3.000	3.500000e+01	3.500000e+01	0.000000e+00
3136	1.000	4.000	6.500000e+01	6.500000e+01	0.000000e+00
3144	1.500	4.000	6.737500e+01	6.737500e+01	0.000000e+00
3152	2.000	4.000	7.200000e+01	7.200000e+01	0.000000e+00

Оценим порядок сходимости:

$$\log_2\left(\frac{\|\widetilde{U}-U(h)\|}{\|\widetilde{U}-U(\frac{h}{2})\|}\right)=\frac{0.0863457}{0.0301859}=1.51625$$

$$\log_2\left(\frac{\|\widetilde{U}-U(\frac{h}{2})\|}{\|\widetilde{U}-U(\frac{h}{4})\|}\right)=\frac{0.0301859}{0.00862528}=1.80723$$

$$\log_2\left(\frac{\|\widetilde{U}-U(\frac{h}{4})\|}{\|\widetilde{U}-U(\frac{h}{8})\|}\right)=\frac{0.00862528}{0.00223976}=1.94523$$

5. Вывод

1.МКР имеет 3-ий порядок апроксимации на равномерных сетках. На не равномерных сетках порядок апроксимации падает до 2-ого

2. На равномерной сетке, как и на не равномерной с 1КУ условиями схема имеет 2-ий порядок сходимости. Сходимость на не равномерной сетке несколько хуже из-за того что погрешность считается по большему шагу, что не позволит в облости с резко меняющимся решением добавить узлов, а в местах малого изменения функции уменьшить количество расчетных узлов.

3. На равномерной сетке с 1-КУ и 2-КУ порядок апроксимации упал до 1-ого, как и порядок сходимости. Это связано с тем, что 2-КУ аппроксимируются с первым порядком, пожтому общий порядок схемы становится первый.

4. Построенная схема не обладает абсолютной устойчивостью, тоесть есть задачи (Например если лямбда большой а гамма маленький), то малое изменение входных параметров приведет к значительному изменению выходных данных.

6. Описание файла CalcArea.txt

* Nx Ny
 x1,1 y1,1 x2,1 y1,2 ... xn,1 y1,n
 x1,2 y2,1 y2,2 y2,2 ... x2,n y2,n
* L - целое число - количество подобластей и далее L наборов чисел по 5 штук
  1 число - номер формул определяющих параметры ДУ в подобласти
  2 число - первая вертикальная ломанная определеяет правую границу
  3 число - вторая вертикальная ломанная определяет левую границу
  их номера определяются в соответствии с правилом для k узла их номера в массивах точек будут: k, (Nx + k), (2*Nx + k), ... , ((Ny-1)*Nx + k)
  4 число - опередеяет горизонтальную границу снизу
  5 число - определяет горизонтальную границу сверху
  Дальше 3 сторики задающие разбиение сетки
* P - целое число - количество подоблостей граничных условий и далее P наборов чисел по 6 штук
  1 число - номер формул определяющих параметры граничных условий
  2 число - Тип краевого условия 0 - соответсвует не заданным краевым
  3 число - первая вертикальная ломанная определеяет правую границу
  4 число - вторая вертикальная ломанная определяет левую границу
  5 число - опередеяет горизонтальную границу снизу
  6 число - определяет горизонтальную границу сверху
* !!! Отличительная особенность одна из координатных линий должна быть фиксированной
* N1 coef1 N2 coef 2 ...
  ...
  ...
  Первый набор задает разбиение каждого отрезка по x
  Второй набора задает разбиение по y

7. Текст программы

main.cpp

```
#include <iostream>
#include "FDM.h"
#include <cmath>


int main()
{
DEquation deq;
BoundCond bound1;
bound1.typeBound = 1;
bound1.func[0] = [](double x,double y) -> double {return x*x*x + y*y*y; };

BoundCond bound2;
bound2.typeBound = 1;
bound2.func[0] = [](double x,double y) -> double {return x*x*x + y*y*y; };

BoundCond bound3;
bound3.typeBound = 1;
bound3.func[0] = [](double x,double y) -> double {return x*x*x + y*y*y; };

BoundCond bound4;
bound4.typeBound = 1;
bound4.func[0] = [](double x,double y) -> double {return x*x*x + y*y*y; };
BoundCond bound5;
bound5.typeBound = 1;
bound5.func[0] = [](double x,double y) -> double {return x*x*x + y*y*y; };

BoundCond bound6;
bound6.typeBound = 1;
bound6.func[0] = [](double x,double y) -> double {return x*x*x + y*y*y; };

BoundCond bound7;
bound7.typeBound = 1;
bound7.func[0] = [](double x,double y) -> double {return x*x*x + y*y*y; };

BoundCond bound8;
bound8.typeBound = 1;
bound8.func[0] = [](double x,double y) -> double {return x*x*x + y*y*y; };

BoundCond bound9;
bound9.typeBound = 1;
bound9.func[0] = [](double x,double y) -> double {return x*x*x + y*y*y; };

BoundCond bound10;
bound10.typeBound = 1;
bound10.func[0] = [](double x,double y) -> double {return x*x*x + y*y*y; };
```

```
deq.f = [](double x, double y) -> double { return -6*x - 6*y + x*x*x + y*y*y;};
deq.gamma = 1;
deq.lambda = 1;
deq.u_true = [](double x, double y) -> double { return x*x*x + y*y*y; };



deq.Bound.resize(10);
deq.Bound[0] = bound1;
deq.Bound[1] = bound2;
deq.Bound[2] = bound3;
deq.Bound[3] = bound4;
deq.Bound[4] = bound5;
deq.Bound[5] = bound6;
deq.Bound[6] = bound7;
deq.Bound[7] = bound8;
deq.Bound[8] = bound9;
deq.Bound[9] = bound10;
FDM fdm("CalcArea1.txt", deq, true);
fdm.Solve();
double norma = fdm.Norma();

cout << "\n\n Норма разности || U* - U || = " << norma << "\n";
fdm.PrintTable();
return 0;
}
```

Grid2D_Quad.h

```
#ifndef GRID2D_QUAD_H_
#define GRID2D_QUAD_H_
#include "Grid.h"
#include "PointInfo.h"
#include <vector>
#include <functional>

using namespace std;

// Сетка представляет из себя решулярное разбиение области при помощи 4-х
угольников произвольной формы
/* Чтение и сетки из файла. Разбиение сетки. */

/* Формат файла
* Nx Ny
* x1,1 y1,1 x2,1 y1,2 ... xn,1 y1,n
* x1,2 y2,1 y2,2 y2,2 ... x2,n y2,n
*
* L - целое число - количество подобластей и далее L наборов чисел по 5 штук
* 1 число - номер формул определяющих параметры ДУ в подобласти
* 2 число - первая вертикальная ломанная определеяет правую границу
```

```
* 3 число - вторая вертикальная ломанная определяет левую границу
их номера определяются в соответствии с правилом для k узла их номера в массивах
точек будут: k, (Nx + k), (2*Nx + k), ... , ( (Ny-1)*Nx + k )
* 4 число - опеределяет горизонтальную границу снизу
* 5 число - определяет горизонтальную границу сверху
* Дальше 3 сторики задающие разбиение сетки
* P - целое число - количество подоблостей граничных условий и далее P наборов
чисел по 6 штук
* 1 число - номер формул определяющих параметры граничных условий
* 2 число - Тип краевого условия 0 - соответсвует не заданным краевым
* 3 число - первая вертикальная ломанная определеяет правую границу
* 4 число - вторая вертикальная ломанная определяет левую границу
* 5 число - опеределяет горизонтальную границу снизу
* 6 число - определяет горизонтальную границу сверху
* !!! Отличительная особенность одна из координатных линий должна быть
фиксированной
* N1 coef1 N2 coef 2 ...
* ...
* ...
* Первый набор задает разбиение каждого отрезка по x
* Второй набора задает разбиение по y
*/

struct BaseGrid2D
{
int CountOfDivision = 1; // Количество делений базовой настройки сетки; 1 -
соответсвует тому что делений не было 2 тому что исходная область разделена в
двое и.т.д
const int SizeOfCalculationAreaElemet = 5;
const int SizeOfBoundsAreaElement = 6;
const int SizeOfDivideParam = 2;

/* Точка */
struct PointXY
{
double x = 0;
double y = 0;
};

struct DivideParamS
{
int num; // количество интервалов на которое нужно разделить отрезок
double coef; // Коэффициент растяжения или сжатия
};

int Nx = 0; // количество узлов вдоль горизонатального направления
int Ny = 0; // количество узлов вдоль вертикального направления
int L = 0; // Количество подоблостей
int P = 0; // Количество видов границ

vector<vector<PointXY>> BaseGridXY; // Базовая сетка в плосткости XY
```

```
// | номер формул | граница по x(start) | граница по x(end) | | граница по y(start) |
граница по y(end) |
vector<vector<int>> CalculationArea; // Массив расчетных областей

//| номер формул | | тип КУ | граница по x(start) | граница по x(end) | граница по
y(start) | граница по y(end) |
vector<vector<int>> BoundsArea;

/*
2 массива
DivideParam[0] - массив для разбиения по оси x размер Nx-1
DivideParam[1] - массив для разбиения по оси y размер Ny-1
*/
vector<vector<DivideParamS>> DivideParam;

/* Флаг того что структура инициализированная и готова к использованию */
// false - Структура не инициализированная
// true - Структура проинициализированна и готова к использованию
bool isReadyToUse = false;
};

struct Point
{
// Информация о границе конкретный набор формул + информация об узле
фиктивный или нет + информация о том граничный ли узел или нет
// + информация о конечном элементе + тип КУ + точка в пространтве
Info info;
double x = 0.0;
double y = 0.0;
};

/* Структура для Конечного Элемента в форме произвольного четырехугольника*/
struct FElement_Quad2D
{

};

/* Структура для шаблона типа Крест для конечно-разностной схемы*/
struct Cross_TypeTemplate
{
static const int CrossTemplateSize = 5; // Количество точек в шаблоне
Point point[CrossTemplateSize]; // Точки шаблона
int AreaInfo = -1; // Номер формул задающий параметры ДУ в данной модификации он
принимает только 1 значение
uint8_t size = 0; // Фактический размер тоесть сколько всего точек в массиве point
[0,5]

/* Информация про главный узел */
/*
isFictive isBound
```

```
0 0 => Просто внутренний узел
0 1 => Не фиктивный и граница ставим апроксимацию краевых в данное место
1 0 => Фиктивный => на диагональ ставим 1
1 1 => не возможно
*/
bool isFictive = false; // Не фиктивный узел
bool isBound = false; // Не граница


/*Если все же это граница то будем инициализировать на */
int ProjNormX = 0; // Проекция внешней нормали для X
int ProjNormY = 0; // Проекция внешней нормали для Y
};


/* Класс сетки */
// Модификация для разностной схемы, но легко модифицируется для МКЭ
class Grid2D_Quad : public GridI<BaseGrid2D,Cross_TypeTemplate>
{

private:
BaseGrid2D baseGrid;

vector<Point> Grid; // Массив точек получающийся при генерации конечных элементов

/* Расчитать общее число узлов получающееся в сетке */
void GetTotalNumberOfNodes() noexcept;
/* Генерация всей расчетной области без учетка фиктивных элементов и принадлежности к какой либо границе и расчетной области */
void GenerateBaseGrid(GridStatus &status) noexcept;

/* Учет фиктивных узлов */
void DivisionIntoSubAreas(GridStatus &status) noexcept;

/* Функция учетка типа КУ и установка факта является ли элемент граничным */
void DivisionIntoSubBounds(GridStatus &status) noexcept;

/* Вернет число соответсвующее стартовой позиции по сути это скачок */
/*
@param
int i - номер
int axis - соответсвующая ость 0 - x, 1 - z 2 - y
@return int: Величина скачка в сетке
*/
int Getlevel(int i, int axis) const noexcept;

protected:
public:
int Dim = 0; // Размерность сетки и СЛАУ в то же время
int GlobalNx = 0; // Сумарное количество узлов по оси X
```

```
int GlobalNy = 0; // Сумарное количество узлов по оси У все величины получаются после генерации сетки

Grid2D_Quad() = default;

// Инициализация класса при помощи файла - формат описан выше
/*
@param
const string &filename - Текстовый файл с разметкой
@return void
Создание объекта
*/
explicit Grid2D_Quad(const string &filename);

// Инициализация "в ручную" при помощи задания структуры
/*
@param
const BaseGrid2D &baseGrid_ - Структура с базовой разметкой области
@return void
Создание объекта
*/
explicit Grid2D_Quad(const BaseGrid2D &baseGrid_);

// Копирование класса отключено
Grid2D_Quad(const Grid2D_Quad &) = delete;
Grid2D_Quad &operator=(const Grid2D_Quad &) = delete;

/* Методы интерфейса обязательны к реализации */

/* Загрузка базовой сетки из файла и по ней строится уже все */
/*
@param
const string &filename - Текстовый файл с разметкой
@return GridStatus
*/
GridStatus Load(const string &filename) noexcept;

/* Генерация сетки */
/*
@param:
void
@return: GridStatus
*/
GridStatus GenerateGrid() noexcept;

/* Дробление сетки в заданное количество раз */
/*
@param:
const int coef - Коэффициент дробления
@return: GridStatus
*/
```

```
GridStatus DivideGrid(const int coef) noexcept;

/* Перегенерация сетки при изменении ее параметров */
/*
@param:
void
@return: GridStatus
*/
GridStatus ReGenerateGrid() noexcept;

/* Гетеры и сеттеры */
/*
@param:
void
@return: BaseGrid2D
*/
BaseGrid2D GetBaseGrid() const noexcept;

/* Получить шаблонный элемент для разностной схемы */
/*
@param:
int idx - Индекс центральной точки в глобальной нумерации
@return: Cross_TypeTemplate - Структура содержащая всю необходимую информацию
*/
Cross_TypeTemplate GetElement(int idx) const noexcept;


/* Получить шаблонный элемент для получения Конечного элемента */
/*
@param:
int idx - Индекс центральной точки в глобальной нумерации
@return: FElement_Quad2D - Структура содержащая всю необходимую информацию
*/
//FElement_Quad2D GetElement(int idx) const noexcept;

/* Установка параметров базовой сетки */
/*
@param:
const BaseGrid2D& baseGrid_ - Базовая сетка области
@return: GridStatus
*/
GridStatus SetBaseGrid(const BaseGrid2D &baseGrid_) noexcept;


Point& operator[](int idx) noexcept;

/* Debug functions */
void PrintBaseGrid() const noexcept;
void PrintGrid() const noexcept;

~Grid2D_Quad() = default;
```

```
};

#endif
```

Grid2D_Quad.cpp

```
#include "Grid2D_Quad.h"
#include "PointInfo.h"
#include <iostream>
#include <cmath>
#include <iomanip>

/* Private section */
int Grid2D_Quad::Getlevel(int i, int axis) const noexcept
{
int res = 0;
for (int k = 0; k < i; k++)
res += baseGrid.DivideParam[axis][k].num;
return res;
};

void Grid2D_Quad::GetTotalNumberOfNodes() noexcept
{
for (int i = 0; i < baseGrid.Nx - 1; i++)
GlobalNx += baseGrid.DivideParam[0][i].num;

for (int i = 0; i < baseGrid.Ny - 1; i++)
GlobalNy += baseGrid.DivideParam[1][i].num;

GlobalNx++;
GlobalNy++;
}

void Grid2D_Quad::GenerateBaseGrid(GridStatus &status) noexcept
{
if (status.GetState() != State::OK)
return;

struct SettingForDivide
{
double step; // Шаг на отрезке
double coef; // Коэффициент увеличения шага
int num; // Количество интервалов идем то num-1 и потом явно вставляем элемент
};
/* Расчитываем шаг для сетки */
/*
@param
int i - Номер массива от 0 до 2
int j - Номер элемента в массиве
```

```
double left - левая грани отрезка
double right - правая граница отрезка
ret: SettingForDivide - структура с вычесленными параметрами деления сетки
*/
auto CalcSettingForDivide = [&](int i, int j, double left, double right) -> SettingForDivide
{
SettingForDivide res;
int num = baseGrid.DivideParam[i][j].num;
double coef = baseGrid.DivideParam[i][j].coef;

if (coef > 1.0)
{
double coefStep = 1.0 + (coef * (std::pow(coef, num - 1) - 1.0)) / (coef - 1.0);

res.step = (right - left) / coefStep;
}
else
{
res.step = (right - left) / num;
}

// Убираем погрешность
if (std::abs(res.step) < eps)
res.step = 0.0;

res.num = num;
res.coef = coef;
return res;
};
/* Генерация разбиения по X или Y( гороизонтальная линия или вертикальная ) с
учетом разбиения */
/*
@param
SettingForDivide &param - параметр разбиения
double left - левая граница отрезка
double right - правая граница отрезка
double *Line - генерируемый массив
int &idx - индекс в массиве на какую позицию ставить элемент
*/
auto GenerateDivide = [](SettingForDivide &param, double left, double right, double *Line,
int &idx) -> void
{
int num = param.num;
double coef = param.coef;
double step = param.step;

Line[idx] = left;
idx++;
double ak = left;
for (int k = 0; k < num - 1; k++)
{
```

```
ak = ak + step * std::pow(coef, k);
Line[idx] = ak;
idx++;
}
Line[idx] = right;
};

try
{
Dim = GlobalNx * GlobalNy;
Grid.resize(Dim);
// Псевдоним для быстрого обращения
int Nx = baseGrid.Nx;
int Ny = baseGrid.Ny;
vector<vector<BaseGrid2D::PointXY>> &BaseGridXY = baseGrid.BaseGridXY;

double *LineX = new double[GlobalNx]; // Массив элементов в строке по X
double *LineY = new double[GlobalNy]; // Массив элементов в строке по Y
// Сгенерируем одну плоскость xy

/* Разбиение по x и y */
/* Расстановка элементов основных линий с учетом их расположения в сетке
( Опорная сетка ) */

/* Расстановка элементов по x ( Опорных ) */

/* Нужно значения x расставить в соответсвующие строки они соответсвуют
разбиению по z */

for (int i = 0; i < Ny; i++)
{
int idx = 0;

for (int j = 0; j < Nx - 1; j++)
{
double left = BaseGridXY[i][j].x;
double right = BaseGridXY[i][j + 1].x;
SettingForDivide param = CalcSettingForDivide(0, j, left, right);
GenerateDivide(param, left, right, LineX, idx);
}
/* Заносим соответствующие значения x на свои позиции */

int startIdx = GetIevel(i, 1) * GlobalNx;
int endIdx = startIdx + GlobalNx;
for (int k = startIdx, kk = 0; k < endIdx; k++, kk++)
Grid[k].x = LineX[kk];
}

/* Расстановка элементов по z ( Опорных ) */
for (int i = 0; i < Nx; i++)
{
```

```
int idx = 0;
for (int j = 0; j < Ny - 1; j++)
{
double left = BaseGridXY[j][i].y;
double right = BaseGridXY[j + 1][i].y;
SettingForDivide param = CalcSettingForDivide(1, j, left, right);
GenerateDivide(param, left, right, LineY, idx);
}

/* Процедура расстановки узлов в глобальный массив */
int startIdx = GetIevel(i, 0); // Стартовый индекс для прохода по массиву
for (int k = 0; k < GlobalNy; k++)
{
// Скачки будут ровно на величину GlobalNx
Grid[startIdx].y = LineY[k];
startIdx += GlobalNx;
}
}

/*****************************************************/

/* Генерация вспомогательных вертикальных линий */
/*
Кратко Алгоритм:
Работаем с осью Y соответственно индексация будет происходить по этой оси
в цикле идем по всем столбцам массива сетки

Нужно получить левую и правую границу на каждом интервале
Сформировать массив отрезков по данной координате
Занести полученный массив в Глобальную сетку
*/
/* Цикл по всем горизонтальным линиям */
for (int i = 0; i < GlobalNx; i++)
{
int idx = 0;
/* Цикл по интеравалам оси Z */
for (int j = 0; j < Ny - 1; j++)
{
int startIdx = i + GlobalNx * GetIevel(j, 1);
int endIdx = i + GlobalNx * GetIevel(j + 1, 1);
double left = Grid[startIdx].x; // Левая граница по x
double right = Grid[endIdx].x; // Правая граница по x

// Разбиение интервала подчиняется разбиению по оси y
SettingForDivide param = CalcSettingForDivide(1, j, left, right);
GenerateDivide(param, left, right, LineY, idx);
}
/* Занесение результата в Итоговый массив */

int startIdx = i; // Стартовая позиция
for (int k = 0; k < GlobalNy; k++)
```

```
{
Grid[startIdx].x = LineY[k];
startIdx += GlobalNx;
}
}
/*********************************************************/

/* Генерация вспомогательных горизонтальных линий */
/* Цикл по всем горизонтальным линиям */
for (int i = 0; i < GlobalNy; i++)
{
int idx = 0;
for (int j = 0; j < Nx - 1; j++)
{
int startIdx = Getlevel(j, 0) + i * GlobalNx;
int endIdx = Getlevel(j + 1, 0) + i * GlobalNx;
double left = Grid[startIdx].y;
double right = Grid[endIdx].y;
// Разбиение интервала подчиняется разбиению по оси x
SettingForDivide param = CalcSettingForDivide(0, j, left, right);
GenerateDivide(param, left, right, LineX, idx);
}
/* Занесение результатов в Глобальную сетку */
int startIdx = i * GlobalNx;
for (int k = 0; k < GlobalNx; k++)
{
Grid[startIdx].y = LineX[k];
startIdx++;
}
}
/*********************************************************/

/* Очистка памяти */
delete[] LineX;
delete[] LineY;
}
catch (std::bad_alloc &e)
{
status.SetStatus(State::MEMORY_ALLOC_ERROR, "Ошибка при выделении памяти в
Grid2D_Quad::GenerateBaseGrid(GridStatus &status)\n");
return;
}
catch (const std::exception &e)
{
status.SetStatus(State::UNKNOWN_ERROR, e.what());
return;
}
}

void Grid2D_Quad::DivisionIntoSubAreas(GridStatus &status) noexcept
{
```

```
if (status.GetState() != State::OK)
return;

struct BoundArea
{
int AreaNum = -1; // номер подобласти (определяет набор формул отвечающих
параметрам ДУ)
int PlaneXZSize = 0; // Размер массива PlaneXZ
int *PlaneXY; // Массив точек многоугольника (номера)
};
int *BoundGrid = nullptr;
try
{
/* Размер массива берем с запасом его размер равен Nx*Ny
контур номеров точек многоугольника соттветствующих каким то координатам
*/
BoundGrid = new int[GlobalNx * GlobalNy];

/* Получает подобласть в соответсвии с ее номером */
/*
@param
int i - Номер подобласти (Порядковый) в массиве определяется порядок
ret BoundArea - сформированный массив подобласти
*/
auto GetBound = [&](int i) -> BoundArea
{
BoundArea Bound;

Bound.AreaNum = baseGrid.CalculationArea[i][0]; // Выставили номер подобласти

/* Вычислим все номера принадлежащие данной области */
/* Инедексы границ многоугольника в глобальной нумерации */
/* XY */

int leftStartX = Getlevel(baseGrid.CalculationArea[i][1], 0) + GlobalNx *
Getlevel(baseGrid.CalculationArea[i][3], 1);
int leftEndX = Getlevel(baseGrid.CalculationArea[i][1], 0) + GlobalNx *
Getlevel(baseGrid.CalculationArea[i][4], 1);
int rightStartX = Getlevel(baseGrid.CalculationArea[i][2], 0) + GlobalNx *
Getlevel(baseGrid.CalculationArea[i][3], 1);
int rightEndX = Getlevel(baseGrid.CalculationArea[i][2], 0) + GlobalNx *
Getlevel(baseGrid.CalculationArea[i][4], 1);


int nX = (rightEndX - rightStartX) / GlobalNx;
int nY = (rightStartX - leftStartX);

int idxBoundGrid = 0;
for (int i = 0; i <= nX; i++)
{
int Idx = leftStartX + i * GlobalNx;
```

```
for (int j = 0; j <= nY; j++)
{
BoundGrid[idxBoundGrid] = Idx;
idxBoundGrid++;
Idx++;
}
}
Bound.PlaneXZSize = idxBoundGrid;
Bound.PlaneXY = BoundGrid;

return Bound;
};
/* Бинарный поиск по массиву */
/*
@param
const BoundArea& Bound - Массив границ
int numPointGlobal - Значение для поиска
ret bool: true - элемент в массиве есть false в противном случае

*/
auto BinarySerch = [](const BoundArea &Bound, int numPointGlobal) -> bool
{
int *arr = Bound.PlaneXY;
int left = 0;
int right = Bound.PlaneXZSize - 1;
int midd = 0;
while (1)
{
midd = (left + right) / 2;

if (numPointGlobal < arr[midd]) // если искомое меньше значения в ячейке
right = midd - 1; // смещаем правую границу поиска
else if (numPointGlobal > arr[midd]) // если искомое больше значения в ячейке
left = midd + 1; // смещаем левую границу поиска
else // иначе (значения равны)
break; // функция возвращает индекс ячейки

// если границы сомкнулись
if (left > right)
{
midd = -1;
break;
}
}
if (midd == -1)
return false;
else
return true;
};

/* Проверяет принадлежит ли точка Заданной области */
```

```
/*
@param
const BoundArea &Bound - Заданная граница
Point &point - Точка для проверки
int numPointGlobal - индекс точки в глобальной нумерации
ret bool: true - Точка принадлежит заданной области, false в противном случае
*/
auto IsInArea = [&](const BoundArea &Bound, Point &point, int numPointGlobal) -> bool
{
auto le = [&](double x1, double x2) -> bool
{
if (x1 < x2)
return true;
if (std::abs(x1 - x2) < eps)
return true;
return false;
};

auto ge = [&](double x1, double x2) -> bool
{
return le(x2, x1);
};

bool arg1 = BinarySerch(Bound, numPointGlobal);
return arg1;
};

/* В цикле по всем областям */
for (int i = 0; i < baseGrid.L; i++)
{
BoundArea Bound = GetBound(i);

/* По той части области где располагаются элементы (включая фиктивный)*/
/* Для этого получим минимальный и максимальный значений индексов */
// int leftAreaBound = ;
// int RightAreaBound = ;

/* Этот цикл тупой он по всем элементам идет */
for (int j = 0; j < Dim; j++)
{
if (IsInArea(Bound, Grid[j], j))
{
/* Мы в заданной области устанавливаем нужные параметры */
InfoManeger::SetFictitious(Grid[j].info, 1);
InfoManeger::SetAreaInfo(Grid[j].info, Bound.AreaNum);
}
}
}

/* Очистка локальной переменной */
delete[] BoundGrid;
```

```
}
catch (std::bad_alloc &e)
{
status.SetStatus(State::MEMORY_ALLOC_ERROR, "Ошибка при выделении памяти в
Grid2D_Quad::GenerateBaseGrid(GridStatus &status)\n");
if (BoundGrid != nullptr)
delete[] BoundGrid;
return;
}
catch (const std::exception &e)
{
status.SetStatus(State::UNKNOWN_ERROR, e.what());
if (BoundGrid != nullptr)
delete[] BoundGrid;
return;
}
}

void Grid2D_Quad::DivisionIntoSubBounds(GridStatus &status) noexcept
{
if (status.GetState() != State::OK)
return;

struct Bound
{
int Size = 0;
int *Line = nullptr; // Массив точек границы
int BoundType = 0;
int BoundFormula = -1;
};

/* Бинарный поиск по массиву */
/*
@param
const BoundArea& Bound - Массив границ
int numPointGlobal - Значение для поиска
ret bool: true - элемент в массиве есть false в противном случае

*/
auto BinarySerch = [](const Bound &bound, int numPointGlobal) -> bool
{
int *arr = bound.Line;
int left = 0;
int right = bound.Size - 1;
int midd = 0;
while (1)
{
midd = (left + right) / 2;

if (numPointGlobal < arr[midd]) // если искомое меньше значения в ячейке
right = midd - 1; // смещаем правую границу поиска
```

```
else if (numPointGlobal > arr[midd]) // если искомое больше значения в ячейке
left = midd + 1; // смещаем левую границу поиска
else // иначе (значения равны)
break; // функция возвращает индекс ячейки

// если границы сомкнулись
if (left > right)
{
midd = -1;
break;
}
}
if (midd == -1)
return false;
else
return true;
};

auto IsInBound = [&](const Bound &bound, int numPointGlobal)
{ return BinarySerch(bound, numPointGlobal); };

int *MemoryPool = nullptr;

try
{

MemoryPool = new int[max(GlobalNx, GlobalNy)];
/* Краевые условия задаются практически так же как и области с тем лишь
исключением, что одна из координат фиксируется */
auto GetBound = [&](int i) -> Bound
{
Bound bound;
bound.BoundType = baseGrid.BoundsArea[i][1];
bound.BoundFormula = baseGrid.BoundsArea[i][0];

/* x - фиксирован */
if (baseGrid.BoundsArea[i][2] == baseGrid.BoundsArea[i][3])
{
/* Определим базовые узлы на прямой OY а потом растиражируем узлы по границе
( Дробление фиксированное ) шаг по массиву GlobalNx*/
int StartPositionY = Getlevel(baseGrid.BoundsArea[i][2], 0) + GlobalNx *
Getlevel(baseGrid.BoundsArea[i][4], 1); // Стартовая позиция для Y
int EndPositionY = Getlevel(baseGrid.BoundsArea[i][2], 0) + GlobalNx *
Getlevel(baseGrid.BoundsArea[i][5], 1); // Конечная точка для оси Y

int nY = (EndPositionY - StartPositionY) / GlobalNx + 1; // Количество узлов по оси Y
int Idx = StartPositionY;
for (int i = 0; i <= nY; i++)
{
MemoryPool[i] = Idx;
Idx += GlobalNx;
```

```
}
bound.Line = MemoryPool;
bound.Size = nY;
}
/* y - фиксирован */
else if (baseGrid.BoundsArea[i][4] == baseGrid.BoundsArea[i][5])
{
/* Определяем базовые узлы по оси OX, а потом аналогично растиражируем узлы по
границе (Дробление фиксированное) шаг по массиву + 1 */
int StartPositionX = Getlevel(baseGrid.BoundsArea[i][2], 0) + GlobalNx *
Getlevel(baseGrid.BoundsArea[i][4], 1); // Стартовая позиция по оси X
int EndPositionX = Getlevel(baseGrid.BoundsArea[i][3], 0) + GlobalNx *
Getlevel(baseGrid.BoundsArea[i][4], 1); // Конечная позиция по оси X
int nX = EndPositionX - StartPositionX + 1; // Количество узлов по оси X
int idxBoundGrid = 0;
int Idx = StartPositionX;
for (int j = 0; j <= nX; j++)
{
MemoryPool[j] = Idx;
Idx++;
}
bound.Line = MemoryPool;
bound.Size = nX;
}

return bound;
};

for (int i = 0; i < baseGrid.P; i++)
{
/* Получить список точек границы */
Bound bound = GetBound(i);
for (int j = 0; j < Dim; j++)
{
/* Расставляем нужные значения в соответствующие точки сетки */
if (IsInBound(bound, j))
{
/* Мы в заданной области устанавливаем нужные параметры */
InfoManeger::SetBoundInfo(Grid[j].info, bound.BoundFormula, bound.BoundType + 1);
}
}
}
}
catch (std::bad_alloc &e)
{
status.SetStatus(State::MEMORY_ALLOC_ERROR, "Ошибка при выделении памяти в
Grid2D_Quad::GenerateBaseGrid(GridStatus &status)\n");
if (MemoryPool != nullptr)
delete[] MemoryPool;
return;
}
```

```
catch (const std::exception &e)
{
status.SetStatus(State::UNKNOWN_ERROR, e.what());
if (MemoryPool != nullptr)
delete[] MemoryPool;
return;
}
}

/*********************************************************/
/* Public section */
Grid2D_Quad::Grid2D_Quad(const string &filename)
{
GridStatus status = Load(filename);
if (status.GetState() != State::OK)
throw status;
}

Grid2D_Quad::Grid2D_Quad(const BaseGrid2D &baseGrid_) : baseGrid(baseGrid_) {}

GridStatus Grid2D_Quad::Load(const string &filename) noexcept
{
GridStatus status;

fin.open(filename);
if (!fin.is_open())
{
status.SetStatus(State::FILE_OPEN_ERROR, "Ошибка открытия файла " + filename + " в
Grid2D_Quad::Grid2D_Quad(const string &filename)\n");
fout.close();
return status;
}

/* Файл корректно открылся и можно читать
Корректность входных данных не проверяется
*/
/* Базовая сетка по XY */
fin >> baseGrid.Nx >> baseGrid.Ny;
baseGrid.BaseGridXY = vector(baseGrid.Ny, vector<BaseGrid2D::PointXY>(baseGrid.Nx));
for (int i = 0; i < baseGrid.Ny; i++)
{
for (int j = 0; j < baseGrid.Nx; j++)
{
fin >> baseGrid.BaseGridXY[i][j].x >> baseGrid.BaseGridXY[i][j].y;
}
}

/* Расчетные подобласти */
fin >> baseGrid.L;
baseGrid.CalculationArea = vector(baseGrid.L,
vector<int>(baseGrid.SizeOfCalculationAreaElemet));
```

```
for (int i = 0; i < baseGrid.L; i++)
{
for (int j = 0; j < baseGrid.SizeOfCalculationAreaElemet; j++)
{
fin >> baseGrid.CalculationArea[i][j];
baseGrid.CalculationArea[i][j]--; // Приведение нумерации с нуля
}
}
/************************/

/* Описание Границ */
fin >> baseGrid.P;
baseGrid.BoundsArea = vector(baseGrid.P,
vector<int>(baseGrid.SizeOfBoundsAreaElement));
for (int i = 0; i < baseGrid.P; i++)
{
for (int j = 0; j < baseGrid.SizeOfBoundsAreaElement; j++)
{
fin >> baseGrid.BoundsArea[i][j];
baseGrid.BoundsArea[i][j]--; // Приведение к нумерации с нуля
}
}
/************************/

/* Правила дробления базовой сетки */
baseGrid.DivideParam.resize(baseGrid.SizeOfDivideParam);
baseGrid.DivideParam[0].resize(baseGrid.Nx - 1);
baseGrid.DivideParam[1].resize(baseGrid.Ny - 1);

for (int i = 0; i < baseGrid.Nx - 1; i++)
{
fin >> baseGrid.DivideParam[0][i].num >> baseGrid.DivideParam[0][i].coef;
}

for (int i = 0; i < baseGrid.Ny - 1; i++)
{
fin >> baseGrid.DivideParam[1][i].num >> baseGrid.DivideParam[1][i].coef;
}

baseGrid.isReadyToUse = true;

return status;
}

GridStatus Grid2D_Quad::GenerateGrid() noexcept
{
GridStatus status;

// Расчет общего количества узлов в сетке
```

```
// Для этого пройдемся по массиву разбиения каждого отрезка и вычислим общее число узлов
GetTotalNumberOfNodes();
// Генерация базовой сетки
GenerateBaseGrid(status);
// Учет фиктивных узлов
DivisionIntoSubAreas(status);

// Учет КУ и расстановка границ
DivisionIntoSubBounds(status);

return status;
}

GridStatus Grid2D_Quad::DivideGrid(const int coef) noexcept
{
GridStatus status;
baseGrid.CountOfDivision *= coef;
for (int i = 0; i < baseGrid.Nx - 1; i++)
{
baseGrid.DivideParam[0][i].num *= coef;
}
for (int i = 0; i < baseGrid.Ny - 1; i++)
{
baseGrid.DivideParam[1][i].num *= coef;
}
return status;
}

GridStatus Grid2D_Quad::ReGenerateGrid() noexcept
{
GridStatus status;
Dim = 0;
GlobalNx = 0;
GlobalNy = 0;
GenerateGrid();
return status;
}

/* Гетеры сеттеры */
BaseGrid2D Grid2D_Quad::GetBaseGrid() const noexcept { return baseGrid; }
Cross_TypeTemplate Grid2D_Quad::GetElement(int idx) const noexcept
{
Cross_TypeTemplate Element;
return Element;
}
GridStatus Grid2D_Quad::SetBaseGrid(const BaseGrid2D &baseGrid_) noexcept
{
GridStatus status;
return status;
}
```

```
Point& Grid2D_Quad::operator[](int idx) noexcept { return Grid[idx]; }

/* Debug functions */
void Grid2D_Quad::PrintBaseGrid() const noexcept
{
cout << "Base Grid Print\n";
cout << "Nx = " << baseGrid.Nx << " Ny = " << baseGrid.Ny << "\n";
for (int i = 0; i < baseGrid.Ny; i++)
{
cout << i + 1 << ": ";
for (int j = 0; j < baseGrid.Nx; j++)
{
cout << "(" << baseGrid.BaseGridXY[i][j].x << ";" << baseGrid.BaseGridXY[i][j].y << ") ";
}
cout << "\n";
}
cout << "\n L = " << baseGrid.L << "\n";
for (int i = 0; i < baseGrid.L; i++)
{
cout << i + 1 << ": ";
for (int j = 0; j < baseGrid.SizeOfCalculationAreaElemet; j++)
{
cout << baseGrid.CalculationArea[i][j] << " ";
}
cout << "\n";
}
cout << "\n P = " << baseGrid.P << "\n";
for (int i = 0; i < baseGrid.P; i++)
{
cout << i + 1 << ": ";
for (int j = 0; j < baseGrid.SizeOfBoundsAreaElement; j++)
{
cout << baseGrid.BoundsArea[i][j] << " ";
}
cout << "\n";
}

cout << "\nDivide Parametrs\n";
cout << "\nDivide X\n";
for (int i = 0; i < baseGrid.Nx - 1; i++)
{
cout << "(num = " << baseGrid.DivideParam[0][i].num << "; coef = " <<
baseGrid.DivideParam[0][i].coef << ")\n";
}
cout << "\nDivide Y\n";
for (int i = 0; i < baseGrid.Ny - 1; i++)
{
cout << "(num = " << baseGrid.DivideParam[1][i].num << "; coef = " <<
baseGrid.DivideParam[1][i].coef << ")\n";
```

```
}
}

void Grid2D_Quad::PrintGrid() const noexcept
{
int idx = 0;
std::cout << "Start idx: " << idx << " End idx: " << GlobalNx * GlobalNy - 1 << " Step
Row: " << GlobalNx << "\n";

for (int i = 0; i < GlobalNy; i++)
{
for (int j = 0; j < GlobalNx; j++)
{
std::cout << std::fixed << std::setprecision(2) << "(" << Grid[idx].x << ";" << Grid[idx].y
<< ") ";
idx++;
}
std::cout << "\n";
}
}

/*****************************************************/
```

SLAU.h

```
#ifndef SLAU_H_
#define SLAU_H_

#include <vector>

using namespace std;

struct Matrix
{
int N = 0; // Размерность матрицы
int m = 0; // Количество не нулевых
int Nd = 2;
vector<vector<double>> ggu;
vector<vector<double>> ggl;
vector<int> ig;
vector<double> di;

vector<int> jOffset;

void SaveMemory();
// умножение матрицы на вектор
std::vector<double> MulMatrVec(const std::vector<double>& x);

/* Вставка элемента на нужно место */
/*
```

```
true - успешно вставили элемент
false - промазали с индексами
*/
bool insert(int i, int j, double val);

double GetElement(int i, int j);

/* Добавить значение в заданную ячейку */
bool add(int i, int j, double val);

/* Распечатка матрицы в плотном вормате */
void PrintDenseFormatMatrix();
};



// Алгоритм работает за константное время так, как нужные индексы однозначно
расчитываются
// и все операции не зависят от размера входных данных
double SumRow(Matrix& matr, std::vector<double>& xk, int i);


std::vector<double> subVec(std::vector<double>& x, const std::vector<double>& y);

// Норма вектора в евклидовом пространстве
double NormVec(const std::vector<double>& vec);



/* Структура для генерации СЛАУ */
struct SLAUData
{
int MAX_ITER = 20e2; // Базовое количество максимального числа итераций
double eps = 1e-14; // Базовое значение невязки
std::vector<double> f;
};


// data - Содержит в себе вектор правой части, макисимальное количество итераций,
заданную точность
// xk1 - начальное приближение и результирующий вектор наше решение кароче
// matr - матрица СЛАУ
void Jakobi(Matrix& matr, std::vector<double>& xk1,SLAUData &data, const double w =
1.0);

// data - Содержит в себе вектор правой части, макисимальное количество итераций,
заданную точность
// xk1 - начальное приближение и результирующий вектор наше решение кароче
// matr - матрица СЛАУ
void Zeidel(Matrix& matr, std::vector<double> &xk1, SLAUData &data, const double w =
1.0);
```

```
#endif
```

SLAU.cpp

```
#include "SLAU.h"
#include <algorithm>
#include <cmath>
#include <iostream>
#include <iomanip>


/* Matrix */
void Matrix::SaveMemory()
{

di.resize(N);
ig.resize(Nd);

ggu.resize(Nd);
ggl.resize(Nd);

jOffset.resize(2 * Nd);

// Нижний треугольник

for (int i = 0; i < Nd; i++)
{
ggl[i].resize(N - 1); // Сразу после главной диагонали

// Верхний треугольник
ggu[i].resize(N - 1); // Сразу после главной диагонали
}

// Составим массив ig
ig[0] = 1;
ig[1] = m + 2;
// Составляем массив оффетов
jOffset = {-2 - m, -1, 1, 2 + m};
}

std::vector<double> Matrix::MulMatrVec(const std::vector<double> &x)
{
std::vector<double> res(N);

for (size_t i = 0; i < (size_t)N; i++)
res[i] = x[i] * di[i];

for (size_t i = 0; i < (size_t)Nd; i++)
```

```
{
for (size_t j = 0; j < (size_t)N - ig[i]; j++)
{
size_t ir = j + ig[i];
res[ir] += ggl[i][j] * x[j];
res[j] += ggu[i][j] * x[ir];
}
}

return res;
}

bool Matrix::add(int i, int j, double val)
{
bool res = true;
std::vector<int>::iterator idx; // Итератор для поиска в списке ig
int k; // Номер строки в массивах ggl, ggu

if( i > j)
{
int ij = i - j;
idx = std::find(ig.begin(), ig.end(), ij);
if (idx != ig.end())
{
k = idx - ig.begin();
ggl[k][j] += val;
}
else
{
res = false;
}
}
else if(i < j)
{
int ji = j - i;
idx = std::find(ig.begin(), ig.end(), ji);
if (idx != ig.end())
{
k = idx - ig.begin();
ggu[k][i] += val;
}
else
{
res = false;
}

}
else
{
di[i] += val;
}
```

```
return res;
}

bool Matrix::insert(int i, int j, double val)
{
bool res = true;
std::vector<int>::iterator idx; // Итератор для поиска в списке ig
int k; // Номер строки в массивах ggl, ggu

if( i > j)
{
int ij = i - j;
idx = std::find(ig.begin(), ig.end(), ij);
if (idx != ig.end())
{
k = idx - ig.begin();
ggl[k][j] = val;
}
else
{
res = false;
}
}
else if(i < j)
{
int ji = j - i;
idx = std::find(ig.begin(), ig.end(), ji);
if (idx != ig.end())
{
k = idx - ig.begin();
ggu[k][i] = val;
}
else
{
res = false;
}

}
else
{
di[i] = val;
}

return res;
}


double Matrix::GetElement(int i, int j)
{
double res = 0;
```

```
std::vector<int>::iterator idx; // Итератор для поиска в списке ig
int k; // Номер строки в массивах ggl, ggu
std::vector<int> j_ = jOffset;
if(i == j)
{
res = di[i];
}
else if( i > j)
{
int ij = i - j;
idx = std::find(ig.begin(), ig.end(), ij);
if (idx != ig.end())
{
k = idx - ig.begin();
res = ggl[k][j];
}
}
else if(i < j)
{
int ji = j - i;
idx = std::find(ig.begin(), ig.end(), ji);
if (idx != ig.end())
{
k = idx - ig.begin();
res = ggu[k][i];
}
}

return res;
}

void Matrix::PrintDenseFormatMatrix()
{
for(int i = 0; i < N; i++)
{
for(int j = 0; j < N; j++)
{
std::cout << std::setw(3) << std::setprecision(3) << GetElement(i,j) << " ";
}
std::cout << "\n";
}
}

/
**********************************************************************************
****/
double SumRow(Matrix &matr, std::vector<double> &xk, int i)
{
double res = matr.di[i] * xk[i];
std::vector<int>::iterator idx; // Итератор для поиска в списке ig
```

```
// Определеяем псевдонимы (ссылки)
std::vector<int> &ig = matr.ig;
std::vector<std::vector<double>> &ggl = matr.ggl;
std::vector<std::vector<double>> &ggu = matr.ggu;

int k; // Номер строки в массивах ggl, ggu

std::vector<int> j_ = matr.jOffset;
for (int k = 0; k < matr.Nd * 2; k++)
j_[k] += i;

for (int j : j_)
{
if (j >= 0)
{
if (i > j)
{
int ij = i - j;
idx = std::find(ig.begin(), ig.end(), ij);
if (idx != ig.end())
{
k = idx - ig.begin();
res += ggl[k][j] * xk[j];
}
}
else if (i < j)
{
int ji = j - i;
idx = std::find(ig.begin(), ig.end(), ji);
if (idx != ig.end())
{
k = idx - ig.begin();
res += ggu[k][i] * xk[j];
}
}
}
}

return res;
}


std::vector<double> subVec(std::vector<double>& x, const std::vector<double>& y)
{
std::vector<double> res(x.size());
for (size_t i = 0; i < x.size(); i++)
res[i] = x[i] - y[i];

return res;
}
```

```
double NormVec(const std::vector<double>& vec)
{
double res = 0.0;
for (size_t i = 0; i < vec.size(); i++)
res += vec[i] * vec[i];

return std::sqrt(res);
}

void Jakobi(Matrix& matr, std::vector<double>& xk1,SLAUData &data, const double w)
{
std::vector<double> xk(matr.N, 0);
int MAX_ITER = data.MAX_ITER;
std::vector<double>& f = data.f;
double eps = data.eps; // Заданная точность
double F_norm = NormVec(data.f); // Норма вектора правой части
double NonRepan; // Относительная невязка
// Итерации идут пока не будет достигнута максимальное число

int n = matr.N;
std::vector<double>& di = matr.di;
for (int k = 0; k < MAX_ITER; k++)
{
for (int i = 0; i < n; i++)
{
xk1[i] = xk[i] + w*(f[i] - SumRow(matr, xk, i)) / matr.di[i];
}
xk = xk1;
// Вычисляем относительную невязку для выхода из цикла по ней
NonRepan = NormVec(subVec(f, matr.MulMatrVec(xk1))) / F_norm;
std::cout << "Iteration = " << k + 1 << " Non-repan = " << NonRepan << "\n";

if (NonRepan < eps) break;
}
}

void Zeidel(Matrix& matr, std::vector<double> &xk1, SLAUData &data, const double w)
{
int n = matr.N;
int MAX_ITER = data.MAX_ITER;
std::vector<double>& f = data.f;
double NonRepan;

double eps = data.eps; // Заданная точность
double F_norm = NormVec(data.f); // Норма вектора правой части
std::vector<double>& di = matr.di;
for (int k = 0; k < MAX_ITER; k++)
{
for (int i = 0; i < n; i++)
{
xk1[i] = xk1[i] + w*(f[i] - SumRow(matr, xk1, i)) / matr.di[i];
```

```
}
NonRepan = NormVec(subVec(f, matr.MulMatrVec(xk1))) / F_norm;
std::cout << "Iteration = " << k+1 << " Non-repan = " << NonRepan << "\n";
if (NonRepan < eps) break; // Выход по неявязке
}
}
```

PointInfo.h

```
#ifndef POINT_INFO_H_
#define POINT_INFO_H_

#include <cstdint>
/* Определеим Структуры и функции для управления информацией о точке в
расчетной области */
/* В общем случае данный класс применим для объектов в 2D и 3D областях */

/* Развертка информации о типе КУ */
struct BoundInfo
{
uint8_t size = 0; // Сколько типов задано
uint8_t Cond[4]{0, 0, 0, 0}; // Набор формул определяющих КУ максимум 2
uint8_t TypeCond[4]{0, 0, 0, 0}; // Тип Краевого условия
};

/* Развертка информации об области */
struct AreaInfo
{
uint8_t size = 0;
uint8_t Cond[8]{0, 0, 0, 0, 0, 0, 0, 0};
};

struct Comand
{
typedef const uint32_t ComandType;
/* Clear comand */
static ComandType BaseInfoClear = 0x00;
static ComandType AreaInfoClear = 0x00000000;
static ComandType BoundInfoClear = 0x0000;
static ComandType TypeBoundCondClear = 0x00;

/* Base Info comand */
static ComandType SetZeroFiFictitious = 0xFE;
static ComandType GetFiFictitious = 0x01;

/* Area Comand */
static ComandType SetZeroAreaCount = 0xE1;
static ComandType GetAreaCount = 0x1E;
static ComandType ShiftAreaCountBits = 1;
static ComandType RightBoundAreaCount = 8;
```

```
static ComandType RightBoundValAreaInfo = 15;
static ComandType BaseAreaShift = 4;
static ComandType GetAreaInfoBaseComand = 0x0000000F;

/* Bound Comand */
static ComandType SetZeroBoundCount = 0x1F;
static ComandType ShiftBoundCountBits = 5;
static ComandType GetBoundCount = 0xE0;
static ComandType RightBoundBoundCount = 4;
static ComandType RightBoundValBoundInfo = 15;
static ComandType RightBoundValBoundType = 3;
static ComandType BaseBoundShift = 4;
static ComandType BaseTypeBoundShift = 2;
static ComandType GetBoundInfoBaseComand = 0x000F;
static ComandType GetTypeBoundBaseComand = 0x03;
};

struct Info
{
uint64_t BaseInfo : 8; /* Базовая информация */
uint64_t AreaInfo : 32; /* Информация об области определяющей параметры ДУ */
uint64_t BoundInfo : 16; /*Краевые условия просто набор формул */
uint64_t TypeBoundCond : 8; /* Тип Краевого условия */
};

class InfoManeger
{
private:

/* Установка Количества различныйх типов областей которым принадлежит точка
@param
Info& info
uint8_t val - диапазон [0;8]
ret void
*/
static void SetAreaCountBits(Info &info, uint32_t val);

/* Установка Количества различных типов границ к которым примыкает данная
точка
@param
Info& info
uint8_t val - диапазон [0;4]
ret void
*/
static void SetBoundCountBits(Info &info, uint32_t val);

/* Получить количество различных областей к которым примыкает точка
@param
const Info &info
ret uint32_t - информациоя о том к скольки областям примыкает диапазон [0;8]
0 - ни к одной
```

```
1 - к одной
...
*/
static uint32_t GetAreaCount(const Info &info);

/* Получить количество различных областей к которым примыкает граница
@param
const Info &info
ret uint32_t - информацияя о том к скольки Границам примыкает(различным)
диапазон [0;4]
0 - ни к какой
1 - к одной
2 -
3 -
4 -
*/
static uint32_t GetBoundCount(const Info &info);

public:

/* Очистка структуры */
/*
@param
Info& info
@return void
Очистка структуры
*/
static void ClearInfo(Info &info);

/* Установка фиктивный/не фиктивный узел
@param
Info& info
uint8_t val true - не фиктивный false - фиктивный
ret void
*/
static void SetFictitious(Info &info, uint8_t val);

/*
@param
Info& info
uint32_t val - набор формул задающий значения на границе диапазон [0,15]
uint32_t boundType - тип Ку диапазон [0,3]
*/
static void SetBoundInfo(Info &info, uint32_t val, uint32_t boundType);

/*
@param
Info& info
uint32_t val - набор формул задающий значения в области определяющей параметры
ДУ диапазон [0,15]
*/
```

```
static void SetAreaInfo(Info &info, uint32_t val);

/*
@param
const Info& info
ret bool - false - фиктивный true - не фиктивный
*/
static bool IsFiFictitious(const Info &info);

/*
@param
const Info& info
ret bool - false - не граница true - граница
*/
static bool IsBound(const Info &info);

/*
@param
const Info& info
ret BoundInfo -структура содержащая информацию о типах Ку и формул которые их
задают
*/
static BoundInfo GetBoundInfo(const Info &info);

/*
@param
const Info& info
ret AreaInfo -структура содержащая информацию о типах областей которым
принадлежит точка и как следствие это определяет парметры ДУ
*/
static AreaInfo GetAreaInfo(const Info &info);
/*****************************************/

/* Debug functions */
static void PrintInfo(const Info &info);

static void PrintBoundInfo(const BoundInfo &Bound);

static void PrintAreaInfo(const AreaInfo &Area);

~InfoManeger() = default;
};

#endif
```

PointInfo.cpp

```
#include "PointInfo.h"
#include <iostream>

/* Privat Section */

void InfoManeger::SetAreaCountBits(Info &info, uint32_t val)
{
if (val <= Comand::RightBoundAreaCount)
{
if (GetAreaCount(info) != 0)
info.BaseInfo &= Comand::SetZeroAreaCount;
info.BaseInfo |= val << Comand::ShiftAreaCountBits;
}
else
throw "Error Bit Operation\n";
}

void InfoManeger::SetBoundCountBits(Info &info, uint32_t val)
{
if (val <= Comand::RightBoundBoundCount)
{
if (GetBoundCount(info) != 0)
info.BaseInfo &= Comand::SetZeroBoundCount;
info.BaseInfo |= val << Comand::ShiftBoundCountBits;
}
else
throw "Error Bit Operation\n";
}

uint32_t InfoManeger::GetAreaCount(const Info &info)
{
return (info.BaseInfo & Comand::GetAreaCount) >> Comand::ShiftAreaCountBits;
}

uint32_t InfoManeger::GetBoundCount(const Info &info)
{
return (info.BaseInfo & Comand::GetBoundCount) >> Comand::ShiftBoundCountBits;
}

/* Public section */

void InfoManeger::ClearInfo(Info &info)
{
info.BaseInfo &= Comand::BaseInfoClear;
info.AreaInfo &= Comand::AreaInfoClear;
info.BoundInfo &= Comand::BoundInfoClear;
info.TypeBoundCond &= Comand::TypeBoundCondClear;
```

```
}


void InfoManeger::SetFictitious(Info &info, uint8_t val)
{
if (val <= 1)
{
if (IsFiFictitious(info))
info.BaseInfo &= Comand::SetZeroFiFictitious;
info.BaseInfo |= val;
}
else
throw "Error Bit Operation\n";
}

void InfoManeger::SetBoundInfo(Info &info, uint32_t val, uint32_t boundType)
{
if (val <= Comand::RightBoundValBoundInfo && boundType <=
Comand::RightBoundValBoundType)
{
uint32_t Bnum = GetBoundCount(info);
/* если не было значений */
if (Bnum == 0)
{
info.BoundInfo |= val;
info.TypeBoundCond |= boundType;
SetBoundCountBits(info, 1);
}
else
{
BoundInfo Bound = GetBoundInfo(info);
for (uint32_t i = 0; i < Bnum; i++)
{
// Проверяем дубликаты
if ((Bound.Cond[i] ^ val) == 0) // Есть дубликат
return;
}

/* Новый элемент */
info.BoundInfo |= val << Comand::BaseBoundShift * Bnum;
info.TypeBoundCond |= boundType << Comand::BaseTypeBoundShift * Bnum;
SetBoundCountBits(info, Bnum + 1);
}
}
else
throw "Error Bit Operation\n";
}

void InfoManeger::SetAreaInfo(Info &info, uint32_t val)
{
if (val <= Comand::RightBoundValAreaInfo)
```

```
{

uint32_t Anum = GetAreaCount(info);
if (Anum == 0)
{
info.AreaInfo |= val;
SetAreaCountBits(info, 1);
}
else
{
AreaInfo Area = GetAreaInfo(info);
for (uint32_t i = 0; i < Anum; i++)
{
// Проверяем дубликаты
if ((Area.Cond[i] ^ val) == 0) // Есть дубликат
return;
}

/* Новый элемент */
info.AreaInfo |= val << Comand::BaseAreaShift * Anum;
SetAreaCountBits(info, Anum + 1);
}
}
else
throw "Error Bit Operation\n";
}

bool InfoManeger::IsFiFictitious(const Info &info)
{
return info.BaseInfo & Comand::GetFiFictitious;
}

bool InfoManeger::IsBound(const Info &info)
{
return GetBoundCount(info) == 0 ? false : true;
}

BoundInfo InfoManeger::GetBoundInfo(const Info &info)
{
BoundInfo Bound;
int32_t Bnum = GetBoundCount(info);
Bound.size = Bnum;

for (int32_t i = 0; i <= Bnum - 1; i++)
{
Bound.Cond[i] = (info.BoundInfo & (Comand::GetBoundInfoBaseComand << i *
Comand::BaseBoundShift)) >> i * Comand::BaseBoundShift;
Bound.TypeCond[i] = (info.TypeBoundCond & (Comand::GetTypeBoundBaseComand << i *
Comand::BaseTypeBoundShift)) >> i * Comand::BaseTypeBoundShift;
}
```

```
return Bound;
}

AreaInfo InfoManeger::GetAreaInfo(const Info &info)
{
AreaInfo Area;
int32_t Anum = GetAreaCount(info);
Area.size = (uint8_t)Anum;

for (int32_t i = 0; i <= Anum - 1; i++)
Area.Cond[i] = (info.AreaInfo & (Comand::GetAreaInfoBaseComand <<
Comand::BaseAreaShift * i)) >> Comand::BaseAreaShift * i;

return Area;
}

void InfoManeger::PrintInfo(const Info &info)
{
std::cout << "Info Struct\n";
std::cout << "IsFiFictitious: " << IsFiFictitious(info) << "\n";
std::cout << "AreaCount: " << GetAreaCount(info) << "\n";
std::cout << "BoundCount: " << GetBoundCount(info) << "\n";
PrintAreaInfo(GetAreaInfo(info));
PrintBoundInfo(GetBoundInfo(info));
}

void InfoManeger::PrintBoundInfo(const BoundInfo &Bound)
{
std::cout << "\nBound Info\n";
for (int i = 0; i < Bound.size; i++)
std::cout << "K = " << i + 1 << " Num formula: " << (uint32_t)Bound.Cond[i] << " Type
Bound: " << (uint32_t)Bound.TypeCond[i] << "\n";
}

void InfoManeger::PrintAreaInfo(const AreaInfo &Area)
{
std::cout << "\nArea Info\n";
for (int i = 0; i < Area.size; i++)
std::cout << "K = " << i + 1 << " Num formula: " << (uint32_t)Area.Cond[i] << "\n";
}
```

GridStatus.h

```
#ifndef GRID_STATUS_H_
#define GRID_STATUS_H_

#include <string>

using namespace std;

enum State
{
OK,
LOAD_ERROR,
FILE_OPEN_ERROR,
MEMORY_ALLOC_ERROR,
GRID_GENERATE_ERROR,
UNKNOWN_ERROR
};

struct Status
{
State state = State::OK;
string msg = "";
};

class GridStatus
{

private:
Status status;

public:
GridStatus() = default;
GridStatus(const Status &status_) : status(status_) {}
inline Status GetStatus() { return status; }
inline string GetMsg() { return status.msg; }
inline State GetState() { return status.state; }
inline void SetStatus(const State& state, const string &msg) { status = {state, msg}; }

~GridStatus() = default;
};

#endif
```

Grid.h

```
#ifndef GRID_H_
#define GRID_H_

#include <string>
#include <fstream>
#include "GridStatus.h"

using namespace std;

template<class BaseGridXD, class ElementXD>
class GridI
{

private:


protected:
double eps = 1e-7; // машинный ноль
ofstream fout; // Файловый поток на запись
ifstream fin; // Файловый поток на чтение

public:

GridI() = default;
/* Загрузка базовой сетки из файла и по ней строится уже все */
virtual GridStatus Load(const string& filename) noexcept = 0;

/* Генерация сетки */
virtual GridStatus GenerateGrid() noexcept = 0;

/* Дробление сетки в заданное количество раз */
virtual GridStatus DivideGrid(const int coef) noexcept = 0;

/* Получить базовую сетку */
virtual BaseGridXD GetBaseGrid() const noexcept = 0;

/* Получить шаблонный элемент который необходим */
virtual ElementXD GetElement(int idx) const noexcept = 0;

/* Перегенерация сетки при изменении ее параметров */
virtual GridStatus ReGenerateGrid() noexcept = 0;
virtual ~GridI() = default;

};

#endif
```

FDM.h

```
#ifndef FDM_H_
#define FDM_H_

#include "Grid2D_Quad.h"
#include "SLAU.h"
#include <tuple>
#include <functional>


/* Шаблон расчетной области - 5-и точечный крест => получается, что краевые узлы
не входят в расчетную область */


struct BoundCond
{
int typeBound = 0; // Тип КУ
function<double(double, double)> func[2];
};

/* Эту структуру инициализируем руками для получения тестов. */
struct DEquation
{
double lambda = 0;
double gamma = 0;
/* Правая часть ДУ */
function<double(double, double)> f = [](double x, double y) -> double { return 0.0; }; // f
= f(x,y)

/* Истинное решение ДУ */
bool IsInitTrue = false; // Имеется ли Истинное решение
function<double(double, double)> u_true;

/* КУ */
vector<BoundCond> Bound;
};


class FDM
{

private:
// false - не равномерная сетка
// true - равномерная сетка
bool IsravGrid = false;
Grid2D_Quad Grid;
Matrix matr;
SLAUData slau;
```

```
vector<double> uij; // Результирующий вектор решений
int N = 0; // размерность СЛАУ
int m = 0; // Количество нулей до диагонали

/* Параметры ДУ */
DEquation deq;
/* Номера гранниц */
int BoundsNodesIdx = 0;
vector<int> BoundsNodes; // Размер 4 * maxSize
/* Генерация Матрицы в которой уже учитывается все необходимое тоесть все КУ */
bool GenerateMatrix();
const int coef = 8;
public:

FDM() = delete;
FDM(const FDM&) = delete;

FDM(const string &filename, DEquation deq_, bool IsravGrid_ = false);

// Решение ДУ
void Solve();

// Расчет расстояния между истинным и расчитанным рещениями считаем как сумма
Квадратов разности
// под Корнем в общем евклидовское расстояние
double Norma();


inline void DivideGrid(const int coef) { Grid.DivideGrid(coef); Grid.ReGenerateGrid(); }

void PrintTable();

FDM& operator=(const FDM&) = delete;

~FDM() = default;

};


#endif
```

FDM.cpp

```
#include "FDM.h"
#include <cmath>
#include <iostream>
#include <stdio.h>

/* Private section */
bool FDM::GenerateMatrix()
{
return true;
}

/*****************************************************************************/

/* Public section */
FDM::FDM(const string &filename, DEquation deq_, bool IsravGrid_) : deq(deq_),
IsravGrid(IsravGrid_)
{
Grid.Load(filename);
Grid.GenerateGrid();

// Расчет m
m = Grid.GlobalNx - 2;

// Сохранение памяти под матрицу
matr.m = m;
matr.N = Grid.Dim;
matr.SaveMemory(); // Есть матрица уже нужной структуры
// Размеры массивов под граничные массивы
BoundsNodes.resize(4 * max(Grid.GlobalNx, Grid.GlobalNy));

//Grid.PrintGrid();
}

void FDM::Solve()
{

uij.clear();
// Очистка того что было, если было
uij.resize(Grid.Dim);
slau.f.resize(Grid.Dim);

// Собираем матрицу для произвольной сетки
if (IsravGrid)
{
double hxRight, hxLeft, hyTop, hyBottom;
double lambda = deq.lambda;
double gamma = deq.gamma;
```

```
for (int i = 0; i < Grid.GlobalNy; i++)
{
for (int j = 0; j < Grid.GlobalNx; j++)
{
int m = i * Grid.GlobalNx + j;
Point centralNode = Grid[m];
if (InfoManeger::IsFiFictitious(centralNode.info))
{
// Узел не фиктивный
if (InfoManeger::IsBound(centralNode.info))
{
// Это граница
BoundInfo bif = InfoManeger::GetBoundInfo(centralNode.info);
int typeCond = (int32_t)bif.TypeCond[0];
// 1 КУ
if (typeCond == 1 || (int)bif.TypeCond[1] == 1)
{
// В рамках задачи на границе может быть только 2 и 1 или 3 и 1 => т.к 1КУ
перекрывают все заносим индекс в массив
BoundsNodes[BoundsNodesIdx] = m;
BoundsNodesIdx++;
// cout << "\n--------------------------------\n";
// cout << "\nNode num = " << m << " Bound Info\n";
// InfoManeger::PrintBoundInfo(bif);
// cout << "--------------------------------\n";
}
else if (typeCond == 2)
{
// Граница S6
if (int(m / Grid.GlobalNx) == 0)
{
hyTop = Grid[m + Grid.GlobalNx].y - Grid[m].y;
// Учет 2 КУ везде это первый элемент на границе S6
matr.add(m, m, lambda / hyTop);
matr.add(m, m + Grid.GlobalNx, -lambda / hyTop);
slau.f[m] += deq.Bound[(int)bif.Cond[0]].func[0](Grid[m].x, Grid[m].y);
//cout << "\nNode num(S6) = " << m << " Bound Info\n";
}
// Граница S3
else
{
hxLeft = Grid[m].x - Grid[m-1].x;
matr.add(m, m, lambda/hxLeft);
matr.add(m, m-1, -lambda/hxLeft);
slau.f[m] += deq.Bound[(int)bif.Cond[0]].func[0](Grid[m].x, Grid[m].y);
//cout << "\nNode num(S3) = " << m << " Bound Info\n";
}
}
else
{
// 3КУ Заносим в матрицу нужные значения
```

```
}

// InfoManeger::PrintInfo(centralNode.info);
// cout << "Node = " << m << " Formula nums = " << (int32_t)bif.Cond[0] << " ";
// cout << "Type Cond = " << << "\n";
}
else
{
// Внутренний узел
int kim1j = m - 1;
int kip1j = m + 1;
int kijm1 = m - Grid.GlobalNx;
int kijp1 = m + Grid.GlobalNx;

hxRight = Grid[kip1j].x - Grid[m].x;
hxLeft = Grid[m].x - Grid[kim1j].x;
hyTop = Grid[kijp1].y - Grid[m].y;
hyBottom = Grid[m].y - Grid[kijm1].y;
// Строка фиксирована это m ая строка

matr.add(m, kim1j, -(2 * lambda) / (hxLeft * (hxLeft + hxRight)));
matr.add(m, kip1j, -(2 * lambda) / (hxRight * (hxLeft + hxRight)));
matr.add(m, kijm1, -(2 * lambda) / (hyBottom * (hyBottom + hyTop)));
matr.add(m, kijp1, -(2 * lambda) / (hyTop * (hyBottom + hyTop)));
matr.add(m, m, lambda * ((2.0) / (hxRight * hxLeft) + (2.0) / (hyTop * hyBottom)) +
gamma);

// Вектор f
slau.f[m] += deq.f(Grid[m].x, Grid[m].y);
}
}
else
{
// Фиктивный узел
slau.f[m] = 0;
matr.insert(m, m, 1.0);
}
}
}

/* А теперь учитываем 1 - ое краевое условие */
for (int i = 0; i < BoundsNodesIdx; i++)
{
int m = BoundsNodes[i];
matr.insert(m, m, 1.0);
BoundInfo bif = InfoManeger::GetBoundInfo(Grid[m].info);
// Вносим значения в праую часть
// cout << "Node Num = " << m << " x = " << Grid[m].x << " y = " << Grid[m].y << "
val = " << deq.Bound[0].func[0](Grid[m].x, Grid[m].y) << "\n";

// Берем от нулевой потому что это всегда 1-КУ
```

```
if((int)bif.TypeCond[0] == 1)
{
slau.f[m] += deq.Bound[(int)bif.Cond[0]].func[0](Grid[m].x, Grid[m].y);
}
else
{
cout << "Node m = " << m << "\n";
slau.f[m] += deq.Bound[(int)bif.Cond[1]].func[0](Grid[m].x, Grid[m].y);
}
}
//matr.PrintDenseFormatMatrix();
//Grid.PrintGrid();
Zeidel(matr, uij, slau, 1.0);
// int k = 0;
// for(double u: uij)
// {
// if(k == Grid.GlobalNx)
// {
// cout << "\n";
// k = 0;
// }
// k++;
// cout << u << " ";
// }
}
else
{
}
}

double FDM::Norma()
{
double res = 0;

/* В цикле по узлам */
for (int i = 0; i < Grid.GlobalNy; i += coef)
{
for (int j = 0; j < Grid.GlobalNx; j += coef)
{
int m = i * Grid.GlobalNx + j;

if (InfoManeger::IsFiFictitious(Grid[m].info))
{
res += pow(deq.u_true(Grid[m].x, Grid[m].y) - uij[m], 2);
}
}
}

return sqrt(res);
}
```

```
void FDM::PrintTable()
{
/* В цикле по узлам */
printf("\n\n Расчетная таблица. Символом * отмечены внутренние узлы сетки\n");
printf("-------------------------------------------------------------------------------------------\n");
printf("|N |X |Y |U |U* | |U* - U| |\n");
printf("|-------------|-------------|-------------|---------------|---------------|---------------|\n");
for (int i = 0; i < Grid.GlobalNy; i+=coef)
{
for (int j = 0; j < Grid.GlobalNx; j+=coef)
{
int m = i * Grid.GlobalNx + j;
if (InfoManeger::IsFiFictitious(Grid[m].info))
{
double u_true = deq.u_true(Grid[m].x, Grid[m].y);
if (InfoManeger::IsBound(Grid[m].info))
{
printf("|%14d|%14.3f|%14.3f| %14e | %14e | %14e |\n", m, Grid[m].x, Grid[m].y, uij[m],
u_true, abs(u_true - uij[m]));
}
else
{
printf("|*%13d|%14.3f|%14.3f| %14e | %14e | %14e |\n", m, Grid[m].x, Grid[m].y, uij[m],
u_true, abs(u_true - uij[m]));
}
}
}
if (i < Grid.GlobalNy - 1)
printf("|-------------|-------------|-------------|---------------|---------------|---------------|\n");
}
printf("-------------------------------------------------------------------------------------------\n\n\n");
}
```